Namespace TeaPie

Classes

Application

ApplicationBuilder

PathExtensions

TeaPie

Interfaces

IApplicationContext

Class Application

Namespace: TeaPie

Assembly: TeaPie.dll

```
public sealed class Application
```

Inheritance

object ← Application

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Methods

Run(CancellationToken)

```
public Task<int> Run(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` CancellationToken

Returns

Task<int>

Class ApplicationBuilder

Namespace: TeaPie

Assembly: TeaPie.dll

```
public sealed class ApplicationBuilder
```

Inheritance

object ← ApplicationBuilder

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Methods

Build()

```
public Application Build()
```

Returns

Application

Create(bool)

```
public static ApplicationBuilder Create(bool collectionRun = true)
```

Parameters

`collectionRun` bool

Returns

ApplicationBuilder

WithDefaultPipeline()

```
public ApplicationBuilder WithDefaultPipeline()
```

Returns

ApplicationBuilder

WithEnvironment(string)

```
public ApplicationBuilder WithEnvironment(string environmentName)
```

Parameters

`environmentName` string

Returns

ApplicationBuilder

WithEnvironmentFile(string)

```
public ApplicationBuilder WithEnvironmentFile(string environmentFilePath)
```

Parameters

`environmentFilePath` string

Returns

ApplicationBuilder

WithInitializationScript(string)

```
public ApplicationBuilder WithInitializationScript(string initializationScriptPath)
```

Parameters

`initializationScriptPath` string

Returns

ApplicationBuilder

WithLogging(LogLevel, string, LogLevel)

```
public ApplicationBuilder WithLogging(LogLevel minimumLevel, string pathToLogFile = "",
LogLevel minimumLevelForLogFile = LogLevel.None)
```

Parameters

`minimumLevel` LogLevel

`pathToLogFile` string

`minimumLevelForLogFile` LogLevel

Returns

ApplicationBuilder

WithPath(string)

```
public ApplicationBuilder WithPath(string path)
```

Parameters

`path` string

Returns

ApplicationBuilder

WithReportFile(string)

```
public ApplicationBuilder WithReportFile(string reportFilePath)
```

Parameters

`reportFilePath` string

Returns

ApplicationBuilder

WithStructureExplorationPipeline()

```
public ApplicationBuilder WithStructureExplorationPipeline()
```

Returns

ApplicationBuilder

WithTemporaryPath(string)

```
public ApplicationBuilder WithTemporaryPath(string temporaryPath)
```

Parameters

`temporaryPath` string

Returns

ApplicationBuilder

Interface IApplicationContext

Namespace: TeaPie

Assembly: TeaPie.dll

```
public interface IApplicationContext
```

Extension Methods
JsonExtensions.ToJsonString(object)

Properties

EnvironmentName

```
string EnvironmentName { get; }
```

Property Value

string

Logger

```
ILogger Logger { get; }
```

Property Value

ILogger

Path

```
string Path { get; }
```

Property Value

string

Reporter

```
ITestResultsSummaryReporter Reporter { get; }
```

Property Value

ITestResultsSummaryReporter

ServiceProvider

```
IServiceProvider ServiceProvider { get; }
```

Property Value

IServiceProvider

Class PathExtensions

Namespace: TeaPie

Assembly: TeaPie.dll

```
public static class PathExtensions
```

Inheritance

object ← PathExtensions

Inherited Members

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

Methods

NormalizePath(string, bool)

```
public static string NormalizePath(this string path, bool rootPath = false)
```

Parameters

`path` string

`rootPath` bool

Returns

string

Class TeaPie

Namespace: TeaPie

Assembly: TeaPie.dll

```
public sealed class TeaPie
```

Inheritance

object ← TeaPie

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object) ,
TeaPieOAuth2Extensions.ConfigureOAuth2Provider(TeaPie, OAuth2Options) ,
TeaPieOAuth2Extensions.SetOAuth2AsDefaultAuthProvider(TeaPie) ,
TeaPieAuthenticationExtensions.RegisterAuthProvider(TeaPie, string, IAuthProvider) ,
TeaPieAuthenticationExtensions.RegisterDefaultProvider(TeaPie, string, IAuthProvider) ,
TeaPieAuthenticationExtensions.SetDefaultAuthProvider(TeaPie, string) ,
TeaPieRetryingExtensions.RegisterRetryStrategy(TeaPie, string,
RetryStrategyOptions<HttpResponseMessage>) ,
TeaPieReportingExtensions.RegisterReporter(TeaPie, Action<TestResultsSummary>) ,
TeaPieReportingExtensions.RegisterReporter(TeaPie, IReporter<TestResultsSummary>) ,
TeaPieTestingExtensions.RegisterTestDirective(TeaPie, string, string, Func<IReadOnlyDictionary<string,
string>, string>, Func<HttpResponseMessage, IReadOnlyDictionary<string, string>, Task>) ,
TeaPieTestingExtensions.Test(TeaPie, string, Action, bool) ,
TeaPieTestingExtensions.Test(TeaPie, string, Func<Task>, bool) ,
TeaPieVariablesExtensions.ContainsVariable(TeaPie, string) ,
TeaPieVariablesExtensions.RemoveVariable(TeaPie, string) ,
TeaPieVariablesExtensions.RemoveVariablesWithTag(TeaPie, string)

Properties

ApplicationContext

```
public IApplicationContext ApplicationContext { get; }
```

Property Value

IApplicationContext

CollectionVariables

```
public VariablesCollection CollectionVariables { get; }
```

Property Value

VariablesCollection

EnvironmentVariables

```
public VariablesCollection EnvironmentVariables { get; }
```

Property Value

VariablesCollection

GlobalVariables

```
public VariablesCollection GlobalVariables { get; }
```

Property Value

VariablesCollection

Instance

```
public static TeaPie? Instance { get; }
```

Property Value

TeaPie

Logger

```
public ILogger Logger { get; }
```

Property Value

ILogger

Request

The most recently executed HTTP request.

```
public HttpRequestMessage? Request { get; }
```

Property Value

HttpRequestMessage

Requests

Collection of current test case's HTTP requests accessible by names.

```
public IReadOnlyDictionary<string, HttpRequestMessage> Requests { get; }
```

Property Value

IReadOnlyDictionary<string, HttpRequestMessage>

Response

The most recently retrieved HTTP response.

```
public HttpResponseMessage? Response { get; }
```

Property Value

HttpResponseMessage

Responses

Collection of current test case's HTTP responses accessible by names.

```
public IReadOnlyDictionary<string, HttpResponseMessage> Responses { get; }
```

Property Value

IReadOnlyDictionary<string, HttpResponseMessage>

TestCaseVariables

```
public VariablesCollection TestCaseVariables { get; }
```

Property Value

VariablesCollection

Methods

GetVariable<T>(string, T?)

Attempts to retrieve the **first matching** variable with the specified `name` of type `T`. If no such variable is found, the `defaultValue` is returned. Variables are searched across all levels in the following order: ***TestCaseVariables, CollectionVariables, EnvironmentVariables, GlobalVariables***.

```
public T? GetVariable<T>(string name, T? defaultValue = default)
```

Parameters

`name` string

The name of the variable to retrieve.

`defaultValue` T

The value to return if no variable with the specified `name` and `T` type is found.

Returns

T

The variable with the specified `name` of type `T`, or `defaultValue` if no matching variable is found.

Type Parameters

`T`

The type of the variable to retrieve.

SetEnvironment(string)

Set environment to one with given `name`. Environment **must be defined in the environment file**.

```
public void SetEnvironment(string name)
```

Parameters

`name` string

Name of the environment to be set.

SetVariable<T>(string, T, params string[])

Stores a variable with the specified `name` of type `T` at the **Collection level**. The variable is tagged with the specified `tags`, which are optional.

```
public void SetVariable<T>(string name, T value, params string[] tags)
```

Parameters

`name` string

The name under which the variable will be stored.

`value` T

The value of the variable to store.

`tags` string[]

An optional list of tags associated with the variable.

Type Parameters

`T`

The type of the variable to store.

Namespace TeaPie.Http

Classes

HttpMessagesExtensions

Class HttpMessagesExtensions

Namespace: TeaPie.Http

Assembly: TeaPie.dll

```
public static class HttpMessagesExtensions
```

Inheritance

object ← HttpMessagesExtensions

Inherited Members

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

Methods

GetBody(HttpRequestMessage)

Gets the body content as a string from the specified `request`.

```
public static string GetBody(this HttpRequestMessage request)
```

Parameters

`request` HttpRequestMessage

The HTTP request message to extract the body content from.

Returns

string

The body content as a string. Returns an **empty string** if the content is null.

GetBody(HttpResponseMessage)

Gets the body content as a string from the specified `response`.

```
public static string GetBody(this HttpResponseMessage response)
```

Parameters

`response` HttpResponseMessage

The HTTP response message to extract the body content from.

Returns

string

The body content as a string. Returns an **empty string** if the content is null.

GetBodyAsExpando(HttpRequestMessage)

Gets the body content as a CaseInsensitiveExpandoObject from the specified `request`. This is possible only if the body content is in JSON structure.

```
public static CaseInsensitiveExpandoObject GetBodyAsExpando(this HttpRequestMessage request)
```

Parameters

`request` HttpRequestMessage

The HTTP request message to extract the body content from.

Returns

CaseInsensitiveExpandoObject

The body content as a CaseInsensitiveExpandoObject.

GetBodyAsExpando(HttpResponseMessage)

Gets the body content as a CaseInsensitiveExpandoObject from the specified `response`. This is possible only if the body content is in JSON structure.

```
public static CaseInsensitiveExpandoObject GetBodyAsExpando(this
HttpResponseMessage response)
```

Parameters

`response` HttpResponseMessage

The HTTP response message to extract the body content from.

Returns

CaseInsensitiveExpandoObject

The body content as a CaseInsensitiveExpandoObject.

GetBodyAsExpandoAsync(HttpRequestMessage)

Asynchronously gets the body content as a CaseInsensitiveExpandoObject from the specified `request`. This is possible only if the body content is in JSON structure.

```
public static Task<CaseInsensitiveExpandoObject> GetBodyAsExpandoAsync(this
HttpRequestMessage request)
```

Parameters

`request` HttpRequestMessage

The HTTP request message to extract the body content from.

Returns

Task<CaseInsensitiveExpandoObject>

A Task that represents the asynchronous operation. The result is the body content as a CaseInsensitiveExpandoObject.

GetBodyAsExpandoAsync(HttpResponseMessage)

Asynchronously gets the body content as a CaseInsensitiveExpandoObject from the specified `response`. This is possible only if the body content is in JSON structure.

```
public static Task<CaseInsensitiveExpandoObject> GetBodyAsExpandoAsync(this
HttpResponseMessage response)
```

Parameters

`response` HttpResponseMessage

The HTTP response message to extract the body content from.

Returns

Task<CaseInsensitiveExpandoObject>

A Task that represents the asynchronous operation. The result is the body content as a CaseInsensitiveExpandoObject.

GetBodyAsync(HttpRequestMessage)

Asynchronously gets the body content as a string from the specified `request`.

```
public static Task<string> GetBodyAsync(this HttpRequestMessage request)
```

Parameters

`request` HttpRequestMessage

The HTTP request message to extract the body content from.

Returns

Task<string>

A Task that represents the asynchronous operation. The result is the body content as a string. Returns an **empty string** if the content is null.

GetBodyAsync(HttpResponseMessage)

Asynchronously gets the body content as a string from the specified `response`.

```
public static Task<string> GetBodyAsync(this HttpResponseMessage response)
```

Parameters

`response` HttpResponseMessage

The HTTP response message to extract the body content from.

Returns

Task<string>

A Task that represents the asynchronous operation. The result is the body content as a string. Returns an **empty string** if the content is null.

GetBodyAsync<TResult>(HttpRequestMessage)

Asynchronously gets the body content as a `TResult` from the specified `request`. This is possible only if the body content is in JSON structure.

```
public static Task<TResult?> GetBodyAsync<TResult>(this HttpRequestMessage request)
```

Parameters

`request` HttpRequestMessage

The HTTP request message to extract the body content from.

Returns

Task<TResult>

A Task that represents the asynchronous operation. The result is the body content as a `TResult`.

Type Parameters

`TResult`

Type which JSON body content will be deserialized to.

GetBodyAsync<TResult>(HttpResponseMessage)

Asynchronously gets the body content as a `TResult` from the specified `response`. This is possible only if the body content is in JSON structure.

```
public static Task<TResult?> GetBodyAsync<TResult>(this HttpResponseMessage response)
```

Parameters

`response` HttpResponseMessage

The HTTP response message to extract the body content from.

Returns

Task<TResult>

A Task that represents the asynchronous operation. The result is the body content as a `TResult`.

Type Parameters

`TResult`

Type which JSON body content will be deserialized to.

GetBody<TResult>(HttpRequestMessage)

Gets the body content as a `TResult` from the specified `request`. This is possible only if the body content is in JSON structure.

```
public static TResult? GetBody<TResult>(this HttpRequestMessage request)
```

Parameters

`request` HttpRequestMessage

The HTTP request message to extract the body content from.

Returns

TResult

The body content as a `TResult`.

Type Parameters

`TResult`

Type which JSON body content will be deserialized to.

GetBody<TResult>(HttpResponseMessage)

Gets the body content as a `TResult` from the specified `response`. This is possible only if the body content is in JSON structure.

```
public static TResult? GetBody<TResult>(this HttpResponseMessage response)
```

Parameters

`response` HttpResponseMessage

The HTTP response message to extract the body content from.

Returns

TResult

The body content as a `TResult`.

Type Parameters

`TResult`

Type which JSON body content will be deserialized to.

StatusCode(HttpResponseMessage)

Gets the status code as an int from the specified `response`.

```
public static int StatusCode(this HttpResponseMessage response)
```

Parameters

`response` HttpResponseMessage

The HTTP response to extract the status code from.

Returns

int

The status code as an int.

Namespace TeaPie.Http.Auth

Classes

TeaPieAuthenticationExtensions

Interfaces

IAuthOptions

IAuthProvider

IAuthProvider<TOptions>

Interface IAuthOptions

Namespace: TeaPie.Http.Auth

Assembly: TeaPie.dll

```
public interface IAuthOptions
```

Extension Methods

JsonExtensions.ToJsonString(object)

Interface IAuthProvider

Namespace: TeaPie.Http.Auth

Assembly: TeaPie.dll

```
public interface IAuthProvider
```

Extension Methods
JsonExtensions.ToJsonString(object)

Methods

Authenticate(HttpRequestMessage, CancellationToken)

```
Task Authenticate(HttpRequestMessage request, CancellationToken cancellationToken)
```

Parameters

`request` HttpRequestMessage

`cancellationToken` CancellationToken

Returns

Task

Interface IAuthProvider<TOptions>

Namespace: TeaPie.Http.Auth

Assembly: TeaPie.dll

```
public interface IAuthProvider<TOptions> : IAuthProvider where TOptions : IAuthOptions
```

Type Parameters

`TOptions`

Inherited Members

IAuthProvider.Authenticate(HttpRequestMessage, CancellationToken)

Extension Methods

JsonExtensions.ToJsonString(object)

Methods

ConfigureOptions(TOptions)

```
IAuthProvider<TOptions> ConfigureOptions(TOptions options)
```

Parameters

`options` TOptions

Returns

IAuthProvider<TOptions>

Class TeaPieAuthenticationExtensions

Namespace: TeaPie.Http.Auth

Assembly: TeaPie.dll

```
public static class TeaPieAuthenticationExtensions
```

Inheritance

object ← TeaPieAuthenticationExtensions

Inherited Members

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

Methods

RegisterAuthProvider(TeaPie, string, IAuthProvider)

Registers an authentication provider with the specified `name`.

```
public static void RegisterAuthProvider(this TeaPie teaPie, string name,
IAuthProvider authenticationProvider)
```

Parameters

`teaPie` TeaPie

The current context instance.

`name` string

The name under which the authentication provider will be registered.

`authenticationProvider` IAuthProvider

The authentication provider to register.

RegisterDefaultProvider(TeaPie, string, IAuthProvider)

Registers an authentication provider with the specified `name` and sets it as the default provider.

```
public static void RegisterDefaultProvider(this TeaPie teaPie, string name,
IAuthProvider authenticationProvider)
```

Parameters

`teaPie` TeaPie

The current context instance.

`name` string

The name under which the authentication provider will be registered.

`authenticationProvider` IAuthProvider

The authentication provider to register.

SetDefaultAuthProvider(TeaPie, string)

Sets the default authentication provider for all requests. A different authentication provider can still be specified for individual requests using a directive.

```
public static void SetDefaultAuthProvider(this TeaPie teaPie, string name)
```

Parameters

`teaPie` TeaPie

The current context instance.

`name` string

The name of a previously registered authentication provider to set as default.

Exceptions

InvalidOperationException

Thrown if no authentication provider is registered with the specified `name`.

Namespace TeaPie.Http.Auth.OAuth2

Classes

OAuth2Options

OAuth2OptionsBuilder

TeaPieOAuth2Extensions

Class OAuth2Options

Namespace: TeaPie.Http.Auth.OAuth2

Assembly: TeaPie.dll

```
public class OAuth2Options : IAuthOptions
```

Inheritance

object ← OAuth2Options

Implements

IAuthOptions

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Properties

AdditionalParameters

```
public IReadOnlyDictionary<string, string> AdditionalParameters { get; }
```

Property Value

IReadOnlyDictionary<string, string>

AuthUrl

```
public string AuthUrl { get; }
```

Property Value

string

ClientId

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

Property Value

string

ClientSecret

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

Property Value

string

GrantType

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

Property Value

string

Password

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

Property Value

string

RedirectUri

```
public Uri? RedirectUri { get; }
```

Property Value

Uri

Username

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

Property Value

string

Class OAuth2OptionsBuilder

Namespace: TeaPie.Http.Auth.OAuth2

Assembly: TeaPie.dll

```
public sealed class OAuth2OptionsBuilder
```

Inheritance

object ← OAuth2OptionsBuilder

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Methods

AddParameter(string, string)

```
public OAuth2OptionsBuilder AddParameter(string key, string value)
```

Parameters

`key` string

`value` string

Returns

OAuth2OptionsBuilder

Build()

```
public OAuth2Options Build()
```

Returns

OAuth2Options

Create()

```
public static OAuth2OptionsBuilder Create()
```

Returns

OAuth2OptionsBuilder

WithAuthUrl(string)

```
public OAuth2OptionsBuilder WithAuthUrl(string oauthUrl)
```

Parameters

`oauthUrl` string

Returns

OAuth2OptionsBuilder

WithClientId(string)

```
public OAuth2OptionsBuilder WithClientId(string clientId)
```

Parameters

`clientId` string

Returns

OAuth2OptionsBuilder

WithClientSecret(string)

```
public OAuth2OptionsBuilder WithClientSecret(string clientSecret)
```

Parameters

`clientSecret` string

Returns

OAuth2OptionsBuilder

WithGrantType(string)

```
public OAuth2OptionsBuilder WithGrantType(string grantType)
```

Parameters

`grantType` string

Returns

OAuth2OptionsBuilder

WithPassword(string)

```
public OAuth2OptionsBuilder WithPassword(string password)
```

Parameters

`password` string

Returns

OAuth2OptionsBuilder

WithUsername(string)

```
public OAuth2OptionsBuilder WithUsername(string username)
```

Parameters

`username` string

Returns

OAuth2OptionsBuilder

Class TeaPieOAuth2Extensions

Namespace: TeaPie.Http.Auth.OAuth2

Assembly: TeaPie.dll

```
public static class TeaPieOAuth2Extensions
```

Inheritance

object ← TeaPieOAuth2Extensions

Inherited Members

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

Methods

ConfigureOAuth2Provider(TeaPie, OAuth2Options)

Configures options for **OAuth2** authentication provider.

```
public static void ConfigureOAuth2Provider(this TeaPie teaPie, OAuth2Options options)
```

Parameters

`teaPie` TeaPie

The current context instance.

`options` OAuth2Options

Options with which **OAuth2** authentication provider will be configured.

SetOAuth2AsDefaultAuthProvider(TeaPie)

Sets **OAuth2** authentication provider as default.

```
public static void SetOAuth2AsDefaultAuthProvider(this TeaPie teaPie)
```

Parameters

`teaPie` TeaPie

The current context instance.

Namespace TeaPie.Http.Retrying

Classes

TeaPieRetryingExtensions

Class TeaPieRetryingExtensions

Namespace: TeaPie.Http.Retrying

Assembly: TeaPie.dll

```
public static class TeaPieRetryingExtensions
```

Inheritance

object ← TeaPieRetryingExtensions

Inherited Members

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

Methods

RegisterRetryStrategy(TeaPie, string, RetryStrategyOptions<HttpResponseMessage>)

Registers `retryStrategy` with given `name`.

```
public static void RegisterRetryStrategy(this TeaPie teaPie, string name,
RetryStrategyOptions<HttpResponseMessage> retryStrategy)
```

Parameters

`teaPie` TeaPie

The current context instance.

`name` string

Name by which retry strategy will be registered.

`retryStrategy` RetryStrategyOptions<HttpResponseMessage>

Retry strategy to be registered.

Namespace TeaPie.Json

Classes

CaseInsensitiveExpandoObject

JsonExtensions

Class CaseInsensitiveExpandoObject

Namespace: TeaPie.Json

Assembly: TeaPie.dll

```
public class CaseInsensitiveExpandoObject : DynamicObject, IDynamicMetaObjectProvider
```

Inheritance

object ← DynamicObject ← CaseInsensitiveExpandoObject

Implements

IDynamicMetaObjectProvider

Inherited Members

DynamicObject.GetMetaObject(Expression) ,
DynamicObject.TryBinaryOperation(BinaryOperationBinder, object, out object) ,
DynamicObject.TryConvert(ConvertBinder, out object) ,
DynamicObject.TryCreateInstance(CreateInstanceBinder, object[], out object) ,
DynamicObject.TryDeleteIndex(DeleteIndexBinder, object[]) ,
DynamicObject.TryDeleteMember(DeleteMemberBinder) ,
DynamicObject.TryGetIndex(GetIndexBinder, object[], out object) ,
DynamicObject.TryInvoke(InvokeBinder, object[], out object) ,
DynamicObject.TryInvokeMember(InvokeMemberBinder, object[], out object) ,
DynamicObject.TrySetIndex(SetIndexBinder, object[], object) ,
DynamicObject.TryUnaryOperation(UnaryOperationBinder, out object) , object.Equals(object) ,
object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

JsonExtensions.ToJsonString(object)

Constructors

CaseInsensitiveExpandoObject(IDictionary<string, object?>)

```
public CaseInsensitiveExpandoObject(IDictionary<string, object?> dictionary)
```

Parameters

`dictionary` IDictionary<string, object>

Methods

GetDynamicMemberNames()

Returns the enumeration of all dynamic member names.

```
public override IEnumerable<string> GetDynamicMemberNames()
```

Returns

IEnumerable<string>

A sequence that contains dynamic member names.

TryGetMember(GetMemberBinder, out object?)

Provides the implementation for operations that get member values. Classes derived from the Dynamic Object class can override this method to specify dynamic behavior for operations such as getting a value for a property.

```
public override bool TryGetMember(GetMemberBinder binder, out object? result)
```

Parameters

`binder` GetMemberBinder

Provides information about the object that called the dynamic operation. The `binder.Name` property provides the name of the member on which the dynamic operation is performed. For example, for the `Console.WriteLine(sampleObject.SampleProperty)` statement, where `sampleObject` is an instance of the class derived from the DynamicObject class, `binder.Name` returns "SampleProperty". The `binder.IgnoreCase` property specifies whether the member name is case-sensitive.

`result` object

The result of the get operation. For example, if the method is called for a property, you can assign the property value to `result`.

Returns

bool

true if the operation is successful; otherwise, false. If this method returns false, the run-time binder of the language determines the behavior. (In most cases, a run-time exception is thrown.)

TrySetMember(SetMemberBinder, object?)

Provides the implementation for operations that set member values. Classes derived from the Dynamic Object class can override this method to specify dynamic behavior for operations such as setting a value for a property.

```
public override bool TrySetMember(SetMemberBinder binder, object? value)
```

Parameters

`binder` SetMemberBinder

Provides information about the object that called the dynamic operation. The `binder.Name` property provides the name of the member to which the value is being assigned. For example, for the statement `sampleObject.SampleProperty = "Test"`, where `sampleObject` is an instance of the class derived from the DynamicObject class, `binder.Name` returns "SampleProperty". The `binder.IgnoreCase` property specifies whether the member name is case-sensitive.

`value` object

The value to set to the member. For example, for `sampleObject.SampleProperty = "Test"`, where `sampleObject` is an instance of the class derived from the DynamicObject class, the `value` is "Test".

Returns

bool

true if the operation is successful; otherwise, false. If this method returns false, the run-time binder of the language determines the behavior. (In most cases, a language-specific run-time exception is thrown.)

Class JsonExtensions

Namespace: TeaPie.Json

Assembly: TeaPie.dll

```
public static class JsonExtensions
```

Inheritance

object ← JsonExtensions

Inherited Members

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

Methods

ToExpando(string)

Parses given text (in JSON structure) to **case-insensitive** expando object (CaseInsensitiveExpandoObject).

```
public static CaseInsensitiveExpandoObject ToExpando(this string jsonText)
```

Parameters

`jsonText` string

Text to be parsed into **case-insensitive** expando object (CaseInsensitiveExpandoObject).

Returns

CaseInsensitiveExpandoObject

CaseInsensitiveExpandoObject representation of JSON stored in `jsonText`.

ToJson(string)

Parses given text to Newtonsoft.Json.Linq.JObject form.

```
public static JObject ToJson(this string text)
```

Parameters

`text` string

Text to be parsed into Newtonsoft.Json.Linq.JObject.

Returns

JObject

Newtonsoft.Json.Linq.JObject representation of JSON within `text`.

ToJsonString(object)

Serializes object to string in **JSON structure**.

```
public static string ToJsonString(this object obj)
```

Parameters

`obj` object

Object that should be serialized to JSON structured string.

Returns

string

string which represents `obj` in **JSON structure**.

To<TResult>(string)

Convert JSON string to `TResult`.

```
public static TResult? To<TResult>(this string jsonText)
```

Parameters

`jsonText` string

String in JSON structure, object should be extracted from.

Returns

TResult

`jsonText` in a `TResult` form.

Type Parameters

`TResult`

Type which JSON string will be deserialized to.

Namespace TeaPie.Reporting

Classes

TeaPieReportingExtensions

Interfaces

ICompositeReporter<TReporterType, TReportedObject>

IReporter

IReporter<TReportedObject>

ITestResultsSummaryReporter

Interface ICompositeReporter<TReporterType, TReportedObject>

Namespace: TeaPie.Reporting

Assembly: TeaPie.dll

```
public interface ICompositeReporter<TReporterType, TReportedObject> : IReporter where
TReporterType : IReporter<TReportedObject>
```

Type Parameters

`TReporterType`

`TReportedObject`

Inherited Members
IReporter.Report()

Extension Methods
JsonExtensions.ToJsonString(object)

Methods

RegisterReporter(TReporterType)

```
void RegisterReporter(TReporterType reporter)
```

Parameters

`reporter` TReporterType

UnregisterReporter(TReporterType)

```
void UnregisterReporter(TReporterType reporter)
```

Parameters

`reporter` TReporterType

Interface IReporter

Namespace: TeaPie.Reporting

Assembly: TeaPie.dll

```
public interface IReporter
```

Extension Methods
JsonExtensions.ToJsonString(object)

Methods

Report()

```
void Report()
```

Interface IReporter<TReportedObject>

Namespace: TeaPie.Reporting

Assembly: TeaPie.dll

```
public interface IReporter<TReportedObject>
```

Type Parameters

`TReportedObject`

Extension Methods
JsonExtensions.ToJsonString(object)

Methods

Report(TReportedObject)

```
void Report(TReportedObject report)
```

Parameters

`report` TReportedObject

Interface ITestResultsSummaryReporter

Namespace: TeaPie.Reporting

Assembly: TeaPie.dll

```
public interface ITestResultsSummaryReporter :
ICompositeReporter<IReporter<TestResultsSummary>, TestResultsSummary>, IReporter
```

Inherited Members

ICompositeReporter<IReporter<TestResultsSummary>, TestResultsSummary>.RegisterReporter(IReporter<TestResultsSummary>) ,
ICompositeReporter<IReporter<TestResultsSummary>, TestResultsSummary>.UnregisterReporter(IReporter<TestResultsSummary>) ,
IReporter.Report()

Extension Methods

JsonExtensions.ToJsonString(object)

Methods

Initialize()

```
void Initialize()
```

RegisterTestResult(string, TestResult)

```
void RegisterTestResult(string testCaseName, TestResult testResult)
```

Parameters

`testCaseName` string

`testResult` TestResult

Class TeaPieReportingExtensions

Namespace: TeaPie.Reporting

Assembly: TeaPie.dll

```
public static class TeaPieReportingExtensions
```

Inheritance

object ← TeaPieReportingExtensions

Inherited Members

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

Methods

RegisterReporter(TeaPie, Action<TestResultsSummary>)

Registers an inline reporter that reports a test results summary using the specified `onReportAction`. The report is generated at the end of the pipeline run.

```
public static void RegisterReporter(this TeaPie teaPie, Action<TestResultsSummary>
onReportAction)
```

Parameters

`teaPie` TeaPie

The current context instance.

`onReportAction` Action<TestResultsSummary>

The action to be executed for test results summary report.

RegisterReporter(TeaPie, IReporter<TestResultsSummary>)

Registers a reporter to generate a test results summary at the end of the pipeline run.

```
public static void RegisterReporter(this TeaPie teaPie, IReporter<TestResultsSummary>
reporter)
```

Parameters

`teaPie` TeaPie

The current context instance.

`reporter` IReporter<TestResultsSummary>

The reporter instance to be added to the collection of reporters.

Namespace TeaPie.Scripts

Classes

Globals

Class Globals

Namespace: TeaPie.Scripts

Assembly: TeaPie.dll

```
public class Globals
```

Inheritance

object ← Globals

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Properties

tp

```
public TeaPie? tp { get; set; }
```

Property Value

TeaPie

Namespace TeaPie.Testing

Classes

CollectionTestResultsSummary

TeaPieTestingExtensions

TestCaseTestResultsSummary

TestDirectivePatternBuilder

A builder for constructing regular expression patterns for test directives used in '.http' files.

TestResult

TestResult.Failed

TestResult.NotRun

TestResult.Passed

TestResultMatchExtensions

TestResultsSummary

Class CollectionTestResultsSummary

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
public class CollectionTestResultsSummary : TestResultsSummary
```

Inheritance

object ← TestResultsSummary ← CollectionTestResultsSummary

Inherited Members

TestResultsSummary.Timestamp , TestResultsSummary.AllTestsPassed , TestResultsSummary.HasSkippedTests , TestResultsSummary.NumberOfSkippedTests , TestResultsSummary.NumberOfPassedTests , TestResultsSummary.NumberOfFailedTests , TestResultsSummary.TimeElapsedDuringTesting , TestResultsSummary.NumberOfTests , TestResultsSummary.NumberOfExecutedTests , TestResultsSummary.PercentageOfSkippedTests , TestResultsSummary.PercentageOfPassedTests , TestResultsSummary.PercentageOfFailedTests , TestResultsSummary.TestResults , TestResultsSummary.SkippedTests , TestResultsSummary.PassedTests , TestResultsSummary.FailedTests , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

JsonExtensions.ToJsonString(object)

Constructors

CollectionTestResultsSummary(string)

```
public CollectionTestResultsSummary(string name = "")
```

Parameters

`name` string

Properties

Name

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

Property Value

string

TestCases

```
public IReadOnlyDictionary<string, TestCaseTestResultsSummary> TestCases { get; }
```

Property Value

IReadOnlyDictionary<string, TestCaseTestResultsSummary>

Class TeaPieTestingExtensions

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
public static class TeaPieTestingExtensions
```

Inheritance

object ← TeaPieTestingExtensions

Inherited Members

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

Methods

RegisterTestDirective(TeaPie, string, string, Func<IReadOnlyDictionary<string, string>, string>, Func<HttpResponseMessage, IReadOnlyDictionary<string, string>, Task>)

Registers a custom test directive that can be used within a '.http' file for any request.

```
public static void RegisterTestDirective(this TeaPie teaPie, string directiveName, string
directivePattern, Func<IReadOnlyDictionary<string, string>, string> testNameGetter,
Func<HttpResponseMessage, IReadOnlyDictionary<string, string>, Task> testFunction)
```

Parameters

`teaPie` TeaPie

The current context instance.

`directiveName` string

The name of the directive, **excluding the 'TEST-' prefix**.

`directivePattern` string

The regular expression pattern for matching the directive. Use TestDirectivePatternBuilder for easier pattern composition.

`testNameGetter` Func<IReadOnlyDictionary<string, string>, string>

A function that generates the full test name. A **dictionary of parameters** is provided for customization of the test name.

`testFunction` Func<HttpResponseMessage, IReadOnlyDictionary<string, string>, Task>

The test function to execute when the directive is applied. The function receives the HTTP response as an HttpResponseMessage and a dictionary of parameters.

Test(TeaPie, string, Action, bool)

Executes the specified `testFunction` as a test method. If `testFunction` throws an exception, the test is considered **failed**. If no exception is thrown, the test is considered **passed**. The test will be referenced by `testName` in the results.

```
public static void Test(this TeaPie teaPie, string testName, Action testFunction, bool
skipTest = false)
```

Parameters

`teaPie` TeaPie

The current context instance.

`testName` string

The name of the test.

`testFunction` Action

The testing function to execute.

`skipTest` bool

Indicates whether the test should be skipped (true) or normally executed (false). Defaults to false.

Test(TeaPie, string, Func<Task>, bool)

Executes the specified asynchronous `testFunction` as a test method. If `testFunction` throws an exception, the test is considered **failed**. If no exception is thrown, the test is considered **passed**. The test will be referenced by `testName` in the results.

```
public static Task Test(this TeaPie teaPie, string testName, Func<Task> testFunction, bool
skipTest = false)
```

Parameters

`teaPie` TeaPie

The current context instance.

`testName` string

The name of the test.

`testFunction` Func<Task>

The asynchronous testing function to execute.

`skipTest` bool

Indicates whether the test should be skipped (true) or normally executed (false). Defaults to false.

Returns

Task

Class TestCaseTestResultsSummary

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
public class TestCaseTestResultsSummary : TestResultsSummary
```

Inheritance

object ← TestResultsSummary ← TestCaseTestResultsSummary

Inherited Members

TestResultsSummary.Timestamp , TestResultsSummary.AllTestsPassed , TestResultsSummary.HasSkippedTests , TestResultsSummary.NumberOfSkippedTests , TestResultsSummary.NumberOfPassedTests , TestResultsSummary.NumberOfFailedTests , TestResultsSummary.TimeElapsedDuringTesting , TestResultsSummary.NumberOfTests , TestResultsSummary.NumberOfExecutedTests , TestResultsSummary.PercentageOfSkippedTests , TestResultsSummary.PercentageOfPassedTests , TestResultsSummary.PercentageOfFailedTests , TestResultsSummary.TestResults , TestResultsSummary.SkippedTests , TestResultsSummary.PassedTests , TestResultsSummary.FailedTests , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Extension Methods

JsonExtensions.ToJsonString(object)

Constructors

TestCaseTestResultsSummary(string)

```
public TestCaseTestResultsSummary(string name)
```

Parameters

`name` string

Properties

Name

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

Property Value

string

Class TestDirectivePatternBuilder

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

A builder for constructing regular expression patterns for test directives used in '.http' files.

```
public sealed class TestDirectivePatternBuilder
```

Inheritance

object ← TestDirectivePatternBuilder

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Methods

AddBooleanArrayParameter(string?)

Adds a parameter that accepts an array of bool.

```
public TestDirectivePatternBuilder AddBooleanArrayParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddBooleanParameter(string?)

Adds a bool parameter.

```
public TestDirectivePatternBuilder AddBooleanParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddDateTimeParameter(string?)

Adds a DateTime parameter.

```
public TestDirectivePatternBuilder AddDateTimeParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddHeaderNameParameter(string?)

Adds a parameter for HTTP header names.

```
public TestDirectivePatternBuilder AddHeaderNameParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddNumberArrayParameter(string?)

Adds a parameter that accepts an array of numbers.

```
public TestDirectivePatternBuilder AddNumberArrayParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddNumberParameter(string?)

Adds a numeric parameter.

```
public TestDirectivePatternBuilder AddNumberParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddParameter(string, string?)

Adds a custom parameter with a specified pattern.

```
public TestDirectivePatternBuilder AddParameter(string pattern, string? parameterName
= null)
```

Parameters

`pattern` string

The regex pattern for the parameter.

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddStatusCodesParameter(string?)

Adds a parameter for HTTP status codes.

```
public TestDirectivePatternBuilder AddStatusCodesParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddStringArrayParameter(string?)

Adds a parameter that accepts an array of string.

```
public TestDirectivePatternBuilder AddStringArrayParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddStringParameter(string?)

Adds a string parameter.

```
public TestDirectivePatternBuilder AddStringParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

AddTimeOnlyParameter(string?)

Adds a TimeOnly parameter.

```
public TestDirectivePatternBuilder AddTimeOnlyParameter(string? parameterName = null)
```

Parameters

`parameterName` string

The optional name of the parameter. If not provided, a default name ("Parameter" + index, starting from 1) is assigned (e.g., "Parameter3").

Returns

TestDirectivePatternBuilder

The updated TestDirectivePatternBuilder instance.

Build()

Builds the final directive pattern as a string.

```
public string Build()
```

Returns

string

The constructed directive pattern.

Create(string)

Creates a new instance of TestDirectivePatternBuilder.

```
public static TestDirectivePatternBuilder Create(string directiveName)
```

Parameters

`directiveName` string

The name of the directive.

Returns

TestDirectivePatternBuilder

A new TestDirectivePatternBuilder instance.

Class TestResult

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
[Union]
public abstract record TestResult : IEquatable<TestResult>
```

Inheritance

object ← TestResult

Implements

IEquatable<TestResult>

Derived

TestResult.Failed, TestResult.NotRun, TestResult.Passed

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Properties

TestName

```
public required string TestName { get; init; }
```

Property Value

string

Methods

Match(Action<NotRun>, Action<Passed>, Action<Failed>)

```
public abstract void Match(Action<TestResult.NotRun> notRun, Action<TestResult.Passed>
passed, Action<TestResult.Failed> failed)
```

Parameters

`notRun` Action<TestResult.NotRun>

`passed` Action<TestResult.Passed>

`failed` Action<TestResult.Failed>

MatchFailed(Action<Failed>, Action)

```
public abstract void MatchFailed(Action<TestResult.Failed> failed, Action @else)
```

Parameters

`failed` Action<TestResult.Failed>

`else` Action

MatchFailed<TMatchOutput>(Func<Failed, TMatchOutput>, Func<TMatchOutput>)

```
public abstract TMatchOutput MatchFailed<TMatchOutput>(Func<TestResult.Failed, TMatchOutput>
failed, Func<TMatchOutput> @else)
```

Parameters

`failed` Func<TestResult.Failed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchFailed<TState>(TState, Action<TState, Failed>, Action<TState>)

```
public abstract void MatchFailed<TState>(TState state, Action<TState, TestResult.Failed>
failed, Action<TState> @else)
```

Parameters

`state` TState

`failed` Action<TState, TestResult.Failed>

`else` Action<TState>

Type Parameters

`TState`

MatchFailed<TState, TMatchOutput>(TState, Func<TState, Failed, TMatchOutput>, Func<TState, TMatchOutput>)

```
public abstract TMatchOutput MatchFailed<TState, TMatchOutput>(TState state, Func<TState,
TestResult.Failed, TMatchOutput> failed, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`failed` Func<TState, TestResult.Failed, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

MatchNotRun(Action<NotRun>, Action)

```
public abstract void MatchNotRun(Action<TestResult.NotRun> notRun, Action @else)
```

Parameters

`notRun` Action<TestResult.NotRun>

`else` Action

MatchNotRun<TMatchOutput>(Func<NotRun, TMatchOutput>, Func<TMatchOutput>)

```
public abstract TMatchOutput MatchNotRun<TMatchOutput>(Func<TestResult.NotRun, TMatchOutput>
notRun, Func<TMatchOutput> @else)
```

Parameters

`notRun` Func<TestResult.NotRun, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchNotRun<TState>(TState, Action<TState, NotRun>, Action<TState>)

```
public abstract void MatchNotRun<TState>(TState state, Action<TState, TestResult.NotRun>
notRun, Action<TState> @else)
```

Parameters

`state` TState

`notRun` Action<TState, TestResult.NotRun>

`else` Action<TState>

Type Parameters

`TState`

MatchNotRun<TState, TMatchOutput>(TState, Func<TState, NotRun, TMatchOutput>, Func<TState, TMatchOutput>)

```
public abstract TMatchOutput MatchNotRun<TState, TMatchOutput>(TState state, Func<TState,
TestResult.NotRun, TMatchOutput> notRun, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`notRun` Func<TState, TestResult.NotRun, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

TMatchOutput

MatchPassed(Action<Passed>, Action)

```
public abstract void MatchPassed(Action<TestResult.Passed> passed, Action @else)
```

Parameters

`passed` Action<TestResult.Passed>

`else` Action

MatchPassed<TMatchOutput>(Func<Passed, TMatchOutput>, Func<TMatchOutput>)

```
public abstract TMatchOutput MatchPassed<TMatchOutput>(Func<TestResult.Passed, TMatchOutput>
passed, Func<TMatchOutput> @else)
```

Parameters

`passed` Func<TestResult.Passed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchPassed<TState>(TState, Action<TState, Passed>, Action<TState>)

```
public abstract void MatchPassed<TState>(TState state, Action<TState, TestResult.Passed>
passed, Action<TState> @else)
```

Parameters

`state` TState

`passed` Action<TState, TestResult.Passed>

`else` Action<TState>

Type Parameters

`TState`

MatchPassed<TState, TMatchOutput>(TState, Func<TState, Passed, TMatchOutput>, Func<TState, TMatchOutput>)

```
public abstract TMatchOutput MatchPassed<TState, TMatchOutput>(TState state, Func<TState,
TestResult.Passed, TMatchOutput> passed, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`passed` Func<TState, TestResult.Passed, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

Match<TMatchOutput>(Func<NotRun, TMatchOutput>, Func<Passed, TMatchOutput>, Func<Failed, TMatchOutput>)

```
public abstract TMatchOutput Match<TMatchOutput>(Func<TestResult.NotRun, TMatchOutput>
notRun, Func<TestResult.Passed, TMatchOutput> passed, Func<TestResult.Failed,
TMatchOutput> failed)
```

Parameters

`notRun` Func<TestResult.NotRun, TMatchOutput>

`passed` Func<TestResult.Passed, TMatchOutput>

`failed` Func<TestResult.Failed, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

Match<TState>(TState, Action<TState, NotRun>, Action<TState, Passed>, Action<TState, Failed>)

```
public abstract void Match<TState>(TState state, Action<TState, TestResult.NotRun> notRun,
Action<TState, TestResult.Passed> passed, Action<TState, TestResult.Failed> failed)
```

Parameters

`state` TState

`notRun` Action<TState, TestResult.NotRun>

`passed` Action<TState, TestResult.Passed>

`failed` Action<TState, TestResult.Failed>

Type Parameters

`TState`

Match<TState, TMatchOutput>(TState, Func<TState, NotRun, TMatchOutput>, Func<TState, Passed, TMatchOutput>, Func<TState, Failed, TMatchOutput>)

```
public abstract TMatchOutput Match<TState, TMatchOutput>(TState state, Func<TState,
TestResult.NotRun, TMatchOutput> notRun, Func<TState, TestResult.Passed, TMatchOutput>
passed, Func<TState, TestResult.Failed, TMatchOutput> failed)
```

Parameters

`state` TState

`notRun` Func<TState, TestResult.NotRun, TMatchOutput>

`passed` Func<TState, TestResult.Passed, TMatchOutput>

`failed` Func<TState, TestResult.Failed, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

UnwrapFailed()

```
public abstract TestResult.Failed UnwrapFailed()
```

Returns

TestResult.Failed

UnwrapNotRun()

```
public abstract TestResult.NotRun UnwrapNotRun()
```

Returns

TestResult.NotRun

UnwrapPassed()

```
public abstract TestResult.Passed UnwrapPassed()
```

Returns

TestResult.Passed

Class TestResult.Failed

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
public sealed record TestResult.Failed : TestResult, IEquatable<TestResult>,
IEquatable<TestResult.Failed>
```

Inheritance

object ← TestResult ← TestResult.Failed

Implements

IEquatable<TestResult>, IEquatable<TestResult.Failed>

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Constructors

Failed(long, string, Exception?)

```
public Failed(long Duration, string ErrorMessage, Exception? Exception)
```

Parameters

`Duration` long

`ErrorMessage` string

`Exception` Exception

Properties

Duration

```
public long Duration { get; init; }
```

Property Value

long

ErrorMessage

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

Property Value

string

Exception

```
public Exception? Exception { get; init; }
```

Property Value

Exception

Methods

Match(Action<NotRun>, Action<Passed>, Action<Failed>)

```
public override void Match(Action<TestResult.NotRun> notRun, Action<TestResult.Passed>
passed, Action<TestResult.Failed> failed)
```

Parameters

notRun Action<TestResult.NotRun>

`passed` Action<TestResult.Passed>

`failed` Action<TestResult.Failed>

MatchFailed(Action<Failed>, Action)

```
public override void MatchFailed(Action<TestResult.Failed> failed, Action @else)
```

Parameters

`failed` Action<TestResult.Failed>

`else` Action

MatchFailed<TMatchOutput>(Func<Failed, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchFailed<TMatchOutput>(Func<TestResult.Failed, TMatchOutput>
failed, Func<TMatchOutput> @else)
```

Parameters

`failed` Func<TestResult.Failed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchFailed<TState>(TState, Action<TState, Failed>, Action<TState>)

```
public override void MatchFailed<TState>(TState state, Action<TState, TestResult.Failed>
failed, Action<TState> @else)
```

Parameters

`state` TState

`failed` Action<TState, TestResult.Failed>

`else` Action<TState>

Type Parameters

`TState`

MatchFailed<TState, TMatchOutput>(TState, Func<TState, Failed, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchFailed<TState, TMatchOutput>(TState state, Func<TState,
TestResult.Failed, TMatchOutput> failed, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`failed` Func<TState, TestResult.Failed, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

MatchNotRun(Action<NotRun>, Action)

```
public override void MatchNotRun(Action<TestResult.NotRun> notRun, Action @else)
```

Parameters

`notRun` Action<TestResult.NotRun>

`else` Action

MatchNotRun<TMatchOutput>(Func<NotRun, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchNotRun<TMatchOutput>(Func<TestResult.NotRun, TMatchOutput>
notRun, Func<TMatchOutput> @else)
```

Parameters

`notRun` Func<TestResult.NotRun, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchNotRun<TState>(TState, Action<TState, NotRun>, Action<TState>)

```
public override void MatchNotRun<TState>(TState state, Action<TState, TestResult.NotRun>
notRun, Action<TState> @else)
```

Parameters

`state` TState

`notRun` Action<TState, TestResult.NotRun>

`else` Action<TState>

Type Parameters

`TState`

MatchNotRun<TState, TMatchOutput>(TState, Func<TState, NotRun, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchNotRun<TState, TMatchOutput>(TState state, Func<TState,
TestResult.NotRun, TMatchOutput> notRun, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`notRun` Func<TState, TestResult.NotRun, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

MatchPassed(Action<Passed>, Action)

```
public override void MatchPassed(Action<TestResult.Passed> passed, Action @else)
```

Parameters

`passed` Action<TestResult.Passed>

`else` Action

MatchPassed<TMatchOutput>(Func<Passed, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchPassed<TMatchOutput>(Func<TestResult.Passed, TMatchOutput>
passed, Func<TMatchOutput> @else)
```

Parameters

`passed` Func<TestResult.Passed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchPassed<TState>(TState, Action<TState, Passed>, Action<TState>)

```
public override void MatchPassed<TState>(TState state, Action<TState, TestResult.Passed>
passed, Action<TState> @else)
```

Parameters

`state` TState

`passed` Action<TState, TestResult.Passed>

`else` Action<TState>

Type Parameters

`TState`

MatchPassed<TState, TMatchOutput>(TState, Func<TState, Passed, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchPassed<TState, TMatchOutput>(TState state, Func<TState,
TestResult.Passed, TMatchOutput> passed, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`passed` Func<TState, TestResult.Passed, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

Match<TMatchOutput>(Func<NotRun, TMatchOutput>, Func<Passed, TMatchOutput>, Func<Failed, TMatchOutput>)

```
public override TMatchOutput Match<TMatchOutput>(Func<TestResult.NotRun, TMatchOutput>
notRun, Func<TestResult.Passed, TMatchOutput> passed, Func<TestResult.Failed,
TMatchOutput> failed)
```

Parameters

`notRun` Func<TestResult.NotRun, TMatchOutput>

`passed` Func<TestResult.Passed, TMatchOutput>

`failed` Func<TestResult.Failed, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

Match<TState>(TState, Action<TState, NotRun>, Action<TState, Passed>, Action<TState, Failed>)

```
public override void Match<TState>(TState state, Action<TState, TestResult.NotRun> notRun,
Action<TState, TestResult.Passed> passed, Action<TState, TestResult.Failed> failed)
```

Parameters

`state` TState

`notRun` Action<TState, TestResult.NotRun>

`passed` Action<TState, TestResult.Passed>

`failed` Action<TState, TestResult.Failed>

Type Parameters

`TState`

Match<TState, TMatchOutput>(TState, Func<TState, NotRun, TMatchOutput>, Func<TState, Passed, TMatchOutput>, Func<TState, Failed, TMatchOutput>)

```
public override TMatchOutput Match<TState, TMatchOutput>(TState state, Func<TState,
TestResult.NotRun, TMatchOutput> notRun, Func<TState, TestResult.Passed, TMatchOutput>
passed, Func<TState, TestResult.Failed, TMatchOutput> failed)
```

Parameters

`state` TState

`notRun` Func<TState, TestResult.NotRun, TMatchOutput>

`passed` Func<TState, TestResult.Passed, TMatchOutput>

`failed` Func<TState, TestResult.Failed, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

UnwrapFailed()

```
public override TestResult.Failed UnwrapFailed()
```

Returns

TestResult.Failed

UnwrapNotRun()

```
public override TestResult.NotRun UnwrapNotRun()
```

Returns

TestResult.NotRun

UnwrapPassed()

```
public override TestResult.Passed UnwrapPassed()
```

Returns

TestResult.Passed

Class TestResult.NotRun

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
public sealed record TestResult.NotRun : TestResult, IEquatable<TestResult>,
IEquatable<TestResult.NotRun>
```

Inheritance

object ← TestResult ← TestResult.NotRun

Implements

IEquatable<TestResult>, IEquatable<TestResult.NotRun>

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Methods

Match(Action<NotRun>, Action<Passed>, Action<Failed>)

```
public override void Match(Action<TestResult.NotRun> notRun, Action<TestResult.Passed>
passed, Action<TestResult.Failed> failed)
```

Parameters

`notRun` Action<TestResult.NotRun>

`passed` Action<TestResult.Passed>

`failed` Action<TestResult.Failed>

MatchFailed(Action<Failed>, Action)

```
public override void MatchFailed(Action<TestResult.Failed> failed, Action @else)
```

Parameters

`failed` Action<TestResult.Failed>

`else` Action

MatchFailed<TMatchOutput>(Func<Failed, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchFailed<TMatchOutput>(Func<TestResult.Failed, TMatchOutput>
failed, Func<TMatchOutput> @else)
```

Parameters

`failed` Func<TestResult.Failed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchFailed<TState>(TState, Action<TState, Failed>, Action<TState>)

```
public override void MatchFailed<TState>(TState state, Action<TState, TestResult.Failed>
failed, Action<TState> @else)
```

Parameters

`state` TState

`failed` Action<TState, TestResult.Failed>

`else` Action<TState>

Type Parameters

`TState`

MatchFailed<TState, TMatchOutput>(TState, Func<TState, Failed, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchFailed<TState, TMatchOutput>(TState state, Func<TState,
TestResult.Failed, TMatchOutput> failed, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`failed` Func<TState, TestResult.Failed, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

MatchNotRun(Action<NotRun>, Action)

```
public override void MatchNotRun(Action<TestResult.NotRun> notRun, Action @else)
```

Parameters

`notRun` Action<TestResult.NotRun>

`else` Action

MatchNotRun<TMatchOutput>(Func<NotRun, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchNotRun<TMatchOutput>(Func<TestResult.NotRun, TMatchOutput>
notRun, Func<TMatchOutput> @else)
```

Parameters

`notRun` Func<TestResult.NotRun, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchNotRun<TState>(TState, Action<TState, NotRun>, Action<TState>)

```
public override void MatchNotRun<TState>(TState state, Action<TState, TestResult.NotRun>
notRun, Action<TState> @else)
```

Parameters

`state` TState

`notRun` Action<TState, TestResult.NotRun>

`else` Action<TState>

Type Parameters

`TState`

MatchNotRun<TState, TMatchOutput>(TState, Func<TState, NotRun, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchNotRun<TState, TMatchOutput>(TState state, Func<TState, TestResult.NotRun, TMatchOutput> notRun, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`notRun` Func<TState, TestResult.NotRun, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

MatchPassed(Action<Passed>, Action)

```
public override void MatchPassed(Action<TestResult.Passed> passed, Action @else)
```

Parameters

`passed` Action<TestResult.Passed>

`else` Action

MatchPassed<TMatchOutput>(Func<Passed, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchPassed<TMatchOutput>(Func<TestResult.Passed, TMatchOutput>
passed, Func<TMatchOutput> @else)
```

Parameters

`passed` Func<TestResult.Passed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchPassed<TState>(TState, Action<TState, Passed>, Action<TState>)

```
public override void MatchPassed<TState>(TState state, Action<TState, TestResult.Passed>
passed, Action<TState> @else)
```

Parameters

`state` TState

`passed` Action<TState, TestResult.Passed>

`else` Action<TState>

Type Parameters

`TState`

MatchPassed<TState, TMatchOutput>(TState, Func<TState, Passed, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchPassed<TState, TMatchOutput>(TState state, Func<TState,
TestResult.Passed, TMatchOutput> passed, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`passed` Func<TState, TestResult.Passed, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

Match<TMatchOutput>(Func<NotRun, TMatchOutput>, Func<Passed, TMatchOutput>, Func<Failed, TMatchOutput>)

```
public override TMatchOutput Match<TMatchOutput>(Func<TestResult.NotRun, TMatchOutput>
notRun, Func<TestResult.Passed, TMatchOutput> passed, Func<TestResult.Failed,
TMatchOutput> failed)
```

Parameters

`notRun` Func<TestResult.NotRun, TMatchOutput>

`passed` Func<TestResult.Passed, TMatchOutput>

`failed` Func<TestResult.Failed, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

Match<TState>(TState, Action<TState, NotRun>, Action<TState, Passed>, Action<TState, Failed>)

```
public override void Match<TState>(TState state, Action<TState, TestResult.NotRun> notRun,
Action<TState, TestResult.Passed> passed, Action<TState, TestResult.Failed> failed)
```

Parameters

`state` TState

`notRun` Action<TState, TestResult.NotRun>

`passed` Action<TState, TestResult.Passed>

`failed` Action<TState, TestResult.Failed>

Type Parameters

`TState`

Match<TState, TMatchOutput>(TState, Func<TState, NotRun, TMatchOutput>, Func<TState, Passed, TMatchOutput>, Func<TState, Failed, TMatchOutput>)

```
public override TMatchOutput Match<TState, TMatchOutput>(TState state, Func<TState,
TestResult.NotRun, TMatchOutput> notRun, Func<TState, TestResult.Passed, TMatchOutput>
passed, Func<TState, TestResult.Failed, TMatchOutput> failed)
```

Parameters

`state` TState

`notRun` Func<TState, TestResult.NotRun, TMatchOutput>

`passed` Func<TState, TestResult.Passed, TMatchOutput>

`failed` Func<TState, TestResult.Failed, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

UnwrapFailed()

```
public override TestResult.Failed UnwrapFailed()
```

Returns

TestResult.Failed

UnwrapNotRun()

```
public override TestResult.NotRun UnwrapNotRun()
```

Returns

TestResult.NotRun

UnwrapPassed()

```
public override TestResult.Passed UnwrapPassed()
```

Returns

TestResult.Passed

Class TestResult.Passed

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
public sealed record TestResult.Passed : TestResult, IEquatable<TestResult>,
IEquatable<TestResult.Passed>
```

Inheritance

object ← TestResult ← TestResult.Passed

Implements

IEquatable<TestResult>, IEquatable<TestResult.Passed>

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Constructors

Passed(long)

```
public Passed(long Duration)
```

Parameters

`Duration` long

Properties

Duration

```
public long Duration { get; init; }
```

Property Value

long

Methods

Match(Action<NotRun>, Action<Passed>, Action<Failed>)

```
public override void Match(Action<TestResult.NotRun> notRun, Action<TestResult.Passed>
passed, Action<TestResult.Failed> failed)
```

Parameters

`notRun` Action<TestResult.NotRun>

`passed` Action<TestResult.Passed>

`failed` Action<TestResult.Failed>

MatchFailed(Action<Failed>, Action)

```
public override void MatchFailed(Action<TestResult.Failed> failed, Action @else)
```

Parameters

`failed` Action<TestResult.Failed>

`else` Action

MatchFailed<TMatchOutput>(Func<Failed, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchFailed<TMatchOutput>(Func<TestResult.Failed, TMatchOutput>
failed, Func<TMatchOutput> @else)
```

Parameters

`failed` Func<TestResult.Failed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchFailed<TState>(TState, Action<TState, Failed>, Action<TState>)

```
public override void MatchFailed<TState>(TState state, Action<TState, TestResult.Failed>
failed, Action<TState> @else)
```

Parameters

`state` TState

`failed` Action<TState, TestResult.Failed>

`else` Action<TState>

Type Parameters

`TState`

MatchFailed<TState, TMatchOutput>(TState, Func<TState, Failed, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchFailed<TState, TMatchOutput>(TState state, Func<TState,
TestResult.Failed, TMatchOutput> failed, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`failed` Func<TState, TestResult.Failed, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

MatchNotRun(Action<NotRun>, Action)

```
public override void MatchNotRun(Action<TestResult.NotRun> notRun, Action @else)
```

Parameters

`notRun` Action<TestResult.NotRun>

`else` Action

MatchNotRun<TMatchOutput>(Func<NotRun, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchNotRun<TMatchOutput>(Func<TestResult.NotRun, TMatchOutput>
notRun, Func<TMatchOutput> @else)
```

Parameters

`notRun` Func<TestResult.NotRun, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchNotRun<TState>(TState, Action<TState, NotRun>, Action<TState>)

```
public override void MatchNotRun<TState>(TState state, Action<TState, TestResult.NotRun>
notRun, Action<TState> @else)
```

Parameters

`state` TState

`notRun` Action<TState, TestResult.NotRun>

`else` Action<TState>

Type Parameters

`TState`

MatchNotRun<TState, TMatchOutput>(TState, Func<TState, NotRun, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchNotRun<TState, TMatchOutput>(TState state, Func<TState,
TestResult.NotRun, TMatchOutput> notRun, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`notRun` Func<TState, TestResult.NotRun, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

MatchPassed(Action<Passed>, Action)

```
public override void MatchPassed(Action<TestResult.Passed> passed, Action @else)
```

Parameters

`passed` Action<TestResult.Passed>

`else` Action

MatchPassed<TMatchOutput>(Func<Passed, TMatchOutput>, Func<TMatchOutput>)

```
public override TMatchOutput MatchPassed<TMatchOutput>(Func<TestResult.Passed, TMatchOutput>
passed, Func<TMatchOutput> @else)
```

Parameters

`passed` Func<TestResult.Passed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

MatchPassed<TState>(TState, Action<TState, Passed>, Action<TState>)

```
public override void MatchPassed<TState>(TState state, Action<TState, TestResult.Passed>
passed, Action<TState> @else)
```

Parameters

`state` TState

`passed` Action<TState, TestResult.Passed>

`else` Action<TState>

Type Parameters

`TState`

MatchPassed<TState, TMatchOutput>(TState, Func<TState, Passed, TMatchOutput>, Func<TState, TMatchOutput>)

```
public override TMatchOutput MatchPassed<TState, TMatchOutput>(TState state, Func<TState,
TestResult.Passed, TMatchOutput> passed, Func<TState, TMatchOutput> @else)
```

Parameters

`state` TState

`passed` Func<TState, TestResult.Passed, TMatchOutput>

`else` Func<TState, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

Match<TMatchOutput>(Func<NotRun, TMatchOutput>, Func<Passed, TMatchOutput>, Func<Failed, TMatchOutput>)

```
public override TMatchOutput Match<TMatchOutput>(Func<TestResult.NotRun, TMatchOutput>
notRun, Func<TestResult.Passed, TMatchOutput> passed, Func<TestResult.Failed,
TMatchOutput> failed)
```

Parameters

`notRun` Func<TestResult.NotRun, TMatchOutput>

`passed` Func<TestResult.Passed, TMatchOutput>

`failed` Func<TestResult.Failed, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TMatchOutput`

Match<TState>(TState, Action<TState, NotRun>, Action<TState, Passed>, Action<TState, Failed>)

```
public override void Match<TState>(TState state, Action<TState, TestResult.NotRun> notRun,
Action<TState, TestResult.Passed> passed, Action<TState, TestResult.Failed> failed)
```

Parameters

`state` TState

`notRun` Action<TState, TestResult.NotRun>

`passed` Action<TState, TestResult.Passed>

`failed` Action<TState, TestResult.Failed>

Type Parameters

`TState`

Match<TState, TMatchOutput>(TState, Func<TState, NotRun, TMatchOutput>, Func<TState, Passed, TMatchOutput>, Func<TState, Failed, TMatchOutput>)

```
public override TMatchOutput Match<TState, TMatchOutput>(TState state, Func<TState,
TestResult.NotRun, TMatchOutput> notRun, Func<TState, TestResult.Passed, TMatchOutput>
passed, Func<TState, TestResult.Failed, TMatchOutput> failed)
```

Parameters

`state` TState

`notRun` Func<TState, TestResult.NotRun, TMatchOutput>

`passed` Func<TState, TestResult.Passed, TMatchOutput>

`failed` Func<TState, TestResult.Failed, TMatchOutput>

Returns

TMatchOutput

Type Parameters

`TState`

`TMatchOutput`

UnwrapFailed()

```
public override TestResult.Failed UnwrapFailed()
```

Returns

TestResult.Failed

UnwrapNotRun()

```
public override TestResult.NotRun UnwrapNotRun()
```

Returns

TestResult.NotRun

UnwrapPassed()

```
public override TestResult.Passed UnwrapPassed()
```

Returns

TestResult.Passed

Class TestResultMatchExtensions

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
public static class TestResultMatchExtensions
```

Inheritance

object ← TestResultMatchExtensions

Inherited Members

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

Methods

MatchAsync(Task<TestResult>, Action<NotRun>, Action<Passed>, Action<Failed>)

```
public static Task MatchAsync(this Task<TestResult> unionTask, Action<TestResult.NotRun>
notRun, Action<TestResult.Passed> passed, Action<TestResult.Failed> failed)
```

Parameters

`unionTask` Task<TestResult>

`notRun` Action<TestResult.NotRun>

`passed` Action<TestResult.Passed>

`failed` Action<TestResult.Failed>

Returns

Task

MatchAsync(ValueTask<TestResult>, Action<NotRun>, Action<Passed>, Action<Failed>)

```
public static ValueTask MatchAsync(this ValueTask<TestResult> unionTask,
Action<TestResult.NotRun> notRun, Action<TestResult.Passed> passed,
Action<TestResult.Failed> failed)
```

Parameters

`unionTask` ValueTask<TestResult>

`notRun` Action<TestResult.NotRun>

`passed` Action<TestResult.Passed>

`failed` Action<TestResult.Failed>

Returns

ValueTask

MatchAsync<TMatchOutput>(Task<TestResult>, Func<NotRun, TMatchOutput>, Func<Passed, TMatchOutput>, Func<Failed, TMatchOutput>)

```
public static Task<TMatchOutput> MatchAsync<TMatchOutput>(this Task<TestResult> unionTask,
Func<TestResult.NotRun, TMatchOutput> notRun, Func<TestResult.Passed, TMatchOutput> passed,
Func<TestResult.Failed, TMatchOutput> failed)
```

Parameters

`unionTask` Task<TestResult>

`notRun` Func<TestResult.NotRun, TMatchOutput>

`passed` Func<TestResult.Passed, TMatchOutput>

`failed` Func<TestResult.Failed, TMatchOutput>

Returns

Task <TMatchOutput>

Type Parameters

`TMatchOutput`

MatchAsync<TMatchOutput>(ValueTask<TestResult>, Func<NotRun, TMatchOutput>, Func<Passed, TMatchOutput>, Func<Failed, TMatchOutput>)

```
public static ValueTask<TMatchOutput> MatchAsync<TMatchOutput>(this ValueTask<TestResult>
unionTask, Func<TestResult.NotRun, TMatchOutput> notRun, Func<TestResult.Passed,
TMatchOutput> passed, Func<TestResult.Failed, TMatchOutput> failed)
```

Parameters

`unionTask` ValueTask <TestResult>

`notRun` Func <TestResult.NotRun, TMatchOutput>

`passed` Func <TestResult.Passed, TMatchOutput>

`failed` Func <TestResult.Failed, TMatchOutput>

Returns

ValueTask <TMatchOutput>

Type Parameters

`TMatchOutput`

MatchFailedAsync(Task<TestResult>, Action<Failed>, Action)

```
public static Task MatchFailedAsync(this Task<TestResult> unionTask,
Action<TestResult.Failed> failed, Action @else)
```

Parameters

`unionTask` Task<TestResult>

`failed` Action<TestResult.Failed>

`else` Action

Returns

Task

MatchFailedAsync(ValueTask<TestResult>, Action<Failed>, Action)

```
public static ValueTask MatchFailedAsync(this ValueTask<TestResult> unionTask,
Action<TestResult.Failed> failed, Action @else)
```

Parameters

`unionTask` ValueTask<TestResult>

`failed` Action<TestResult.Failed>

`else` Action

Returns

ValueTask

MatchFailedAsync<TMatchOutput>(Task<TestResult>, Func<Failed, TMatchOutput>, Func<TMatchOutput>)

```
public static Task<TMatchOutput> MatchFailedAsync<TMatchOutput>(this Task<TestResult>
unionTask, Func<TestResult.Failed, TMatchOutput> failed, Func<TMatchOutput> @else)
```

Parameters

`unionTask` Task<TestResult>

`failed` Func<TestResult.Failed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

Task<TMatchOutput>

Type Parameters

`TMatchOutput`

MatchFailedAsync<TMatchOutput>(ValueTask<TestResult>, Func<Failed, TMatchOutput>, Func<TMatchOutput>)

```
public static ValueTask<TMatchOutput> MatchFailedAsync<TMatchOutput>(this
ValueTask<TestResult> unionTask, Func<TestResult.Failed, TMatchOutput> failed,
Func<TMatchOutput> @else)
```

Parameters

`unionTask` ValueTask<TestResult>

`failed` Func<TestResult.Failed, TMatchOutput>

`else` Func<TMatchOutput>

Returns

ValueTask<TMatchOutput>

Type Parameters

`TMatchOutput`

MatchNotRunAsync(Task<TestResult>, Action<NotRun>,

Action)

```
public static Task MatchNotRunAsync(this Task<TestResult> unionTask,
Action<TestResult.NotRun> notRun, Action @else)
```

Parameters

`unionTask` Task<TestResult>

`notRun` Action<TestResult.NotRun>

`else` Action

Returns

Task

MatchNotRunAsync(ValueTask<TestResult>, Action<NotRun>, Action)

```
public static ValueTask MatchNotRunAsync(this ValueTask<TestResult> unionTask,
Action<TestResult.NotRun> notRun, Action @else)
```

Parameters

`unionTask` ValueTask<TestResult>

`notRun` Action<TestResult.NotRun>

`else` Action

Returns

ValueTask

MatchNotRunAsync<TMatchOutput>(Task<TestResult>, Func<NotRun, TMatchOutput>, Func<TMatchOutput>)

```
public static Task<TMatchOutput> MatchNotRunAsync<TMatchOutput>(this Task<TestResult>
unionTask, Func<TestResult.NotRun, TMatchOutput> notRun, Func<TMatchOutput> @else)
```

Parameters

`unionTask` Task<TestResult>

`notRun` Func<TestResult.NotRun, TMatchOutput>

`else` Func<TMatchOutput>

Returns

Task<TMatchOutput>

Type Parameters

`TMatchOutput`

MatchNotRunAsync<TMatchOutput>(ValueTask<TestResult>, Func<NotRun, TMatchOutput>, Func<TMatchOutput>)

```
public static ValueTask<TMatchOutput> MatchNotRunAsync<TMatchOutput>(this
ValueTask<TestResult> unionTask, Func<TestResult.NotRun, TMatchOutput> notRun,
Func<TMatchOutput> @else)
```

Parameters

`unionTask` ValueTask<TestResult>

`notRun` Func<TestResult.NotRun, TMatchOutput>

`else` Func<TMatchOutput>

Returns

ValueTask<TMatchOutput>

Type Parameters

`TMatchOutput`

MatchPassedAsync(Task<TestResult>, Action<Passed>, Action)

```
public static Task MatchPassedAsync(this Task<TestResult> unionTask,
Action<TestResult.Passed> passed, Action @else)
```

Parameters

`unionTask` Task<TestResult>

`passed` Action<TestResult.Passed>

`else` Action

Returns

Task

MatchPassedAsync(ValueTask<TestResult>, Action<Passed>, Action)

```
public static ValueTask MatchPassedAsync(this ValueTask<TestResult> unionTask,
Action<TestResult.Passed> passed, Action @else)
```

Parameters

`unionTask` ValueTask<TestResult>

`passed` Action<TestResult.Passed>

`else` Action

Returns

ValueTask

MatchPassedAsync<TMatchOutput>(Task<TestResult>, Func<Passed, TMatchOutput>, Func<TMatchOutput>)

```
public static Task<TMatchOutput> MatchPassedAsync<TMatchOutput>(this Task<TestResult>
unionTask, Func<TestResult.Passed, TMatchOutput> passed, Func<TMatchOutput> @else)
```

Parameters

`unionTask` Task <TestResult>

`passed` Func <TestResult.Passed, TMatchOutput>

`else` Func <TMatchOutput>

Returns

Task <TMatchOutput>

Type Parameters

`TMatchOutput`

MatchPassedAsync<TMatchOutput>(ValueTask<TestResult>, Func<Passed, TMatchOutput>, Func<TMatchOutput>)

```
public static ValueTask<TMatchOutput> MatchPassedAsync<TMatchOutput>(this
ValueTask<TestResult> unionTask, Func<TestResult.Passed, TMatchOutput> passed,
Func<TMatchOutput> @else)
```

Parameters

`unionTask` ValueTask <TestResult>

`passed` Func <TestResult.Passed, TMatchOutput>

`else` Func <TMatchOutput>

Returns

ValueTask <TMatchOutput>

Type Parameters

`TMatchOutput`

Class TestResultsSummary

Namespace: TeaPie.Testing

Assembly: TeaPie.dll

```
public class TestResultsSummary
```

Inheritance

object ← TestResultsSummary

Derived

CollectionTestResultsSummary, TestCaseTestResultsSummary

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Properties

AllTestsPassed

```
public bool AllTestsPassed { get; }
```

Property Value

bool

FailedTests

```
public IReadOnlyList<TestResult.Failed> FailedTests { get; }
```

Property Value

IReadOnlyList<TestResult.Failed>

HasSkippedTests

```
public bool HasSkippedTests { get; }
```

Property Value

bool

NumberOfExecutedTests

```
public int NumberOfExecutedTests { get; }
```

Property Value

int

NumberOfFailedTests

```
public int NumberOfFailedTests { get; }
```

Property Value

int

NumberOfPassedTests

```
public int NumberOfPassedTests { get; }
```

Property Value

int

NumberOfSkippedTests

```
public int NumberOfSkippedTests { get; }
```

Property Value

int

NumberOfTests

```
public int NumberOfTests { get; }
```

Property Value

int

PassedTests

```
public IReadOnlyList<TestResult.Passed> PassedTests { get; }
```

Property Value

IReadOnlyList <TestResult.Passed>

PercentageOfFailedTests

```
public double PercentageOfFailedTests { get; }
```

Property Value

double

PercentageOfPassedTests

```
public double PercentageOfPassedTests { get; }
```

Property Value

double

PercentageOfSkippedTests

```
public double PercentageOfSkippedTests { get; }
```

Property Value

double

SkippedTests

```
public IReadOnlyList<TestResult.NotRun> SkippedTests { get; }
```

Property Value

IReadOnlyList<TestResult.NotRun>

TestResults

```
public IReadOnlyList<TestResult> TestResults { get; }
```

Property Value

IReadOnlyList<TestResult>

TimeElapsedDuringTesting

```
public double TimeElapsedDuringTesting { get; }
```

Property Value

double

Timestamp

```
public DateTime Timestamp { get; protected set; }
```

Property Value

DateTime

Namespace TeaPie.Variables

Classes

TeaPieVariablesExtensions

Variable

VariablesCollection

Interfaces

IVariablesOperations

Interface IVariablesOperations

Namespace: TeaPie.Variables

Assembly: TeaPie.dll

```
public interface IVariablesOperations
```

Extension Methods
JsonExtensions.ToJsonString(object)

Methods

ContainsVariable(string)

```
bool ContainsVariable(string name)
```

Parameters

`name` string

Returns

bool

GetVariable<T>(string, T?)

```
T? GetVariable<T>(string name, T? defaultValue = default)
```

Parameters

`name` string

`defaultValue` T

Returns

T

Type Parameters

T

RemoveVariable(string)

```
bool RemoveVariable(string name)
```

Parameters

name string

Returns

bool

RemoveVariablesWithTag(string)

```
bool RemoveVariablesWithTag(string tag)
```

Parameters

tag string

Returns

bool

SetVariable<T>(string, T, params string[])

```
void SetVariable<T>(string name, T value, params string[] tags)
```

Parameters

`name` string

`value` T

`tags` string[]

Type Parameters

`T`

Class TeaPieVariablesExtensions

Namespace: TeaPie.Variables

Assembly: TeaPie.dll

```
public static class TeaPieVariablesExtensions
```

Inheritance

object ← TeaPieVariablesExtensions

Inherited Members

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

Methods

ContainsVariable(TeaPie, string)

Determines whether a variable with the specified `name` exists.

```
public static bool ContainsVariable(this TeaPie teaPie, string name)
```

Parameters

`teaPie` TeaPie

The current context instance.

`name` string

The name of the variable to check for existence.

Returns

bool

`true` if a variable with the specified `name` exists; otherwise, `false`.

RemoveVariable(TeaPie, string)

Attempts to remove the variable(s) with the specified `name` from all levels (***TestCaseVariables, CollectionVariables, EnvironmentVariables, GlobalVariables***).

```
public static bool RemoveVariable(this TeaPie teaPie, string name)
```

Parameters

`teaPie` TeaPie

The current context instance.

`name` string

The name of the variable(s) to remove.

Returns

bool

`true` if the variable(s) were successfully removed from all levels; otherwise, `false`.

RemoveVariablesWithTag(TeaPie, string)

Attempts to remove all variables tagged with the specified `tag` from all levels (***TestCaseVariables, CollectionVariables, EnvironmentVariables, GlobalVariables***).

```
public static bool RemoveVariablesWithTag(this TeaPie teaPie, string tag)
```

Parameters

`teaPie` TeaPie

The current context instance.

`tag` string

The tag used to identify variables for removal.

Returns

bool

`true` if all variables with the specified tag were successfully removed from all levels; otherwise, `false`.

Class Variable

Namespace: TeaPie.Variables

Assembly: TeaPie.dll

```
public class Variable
```

Inheritance

object ← Variable

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Constructors

Variable(string, object?, params string[])

```
public Variable(string name, object? value, params string[] tags)
```

Parameters

`name` string

`value` object

`tags` string[]

Properties

Name

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

Property Value

string

Methods

GetValue<T>()

```
public T? GetValue<T>()
```

Returns

T

Type Parameters

T

Class VariablesCollection

Namespace: TeaPie.Variables

Assembly: TeaPie.dll

```
public class VariablesCollection : IEnumerable<Variable>, IEnumerable
```

Inheritance

object ← VariablesCollection

Implements

IEnumerable<Variable>, IEnumerable

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Properties

Count

```
public int Count { get; }
```

Property Value

int

Methods

Clear()

```
public void Clear()
```

Contains(string)

```
public bool Contains(string variableName)
```

Parameters

`variableName` string

Returns

bool

GetEnumerator()

Returns an enumerator that iterates through the collection.

```
public IEnumerator<Variable> GetEnumerator()
```

Returns

IEnumerator <Variable>

An enumerator that can be used to iterate through the collection.

Get<T>(string, T?)

Attempts to get variable with given `name` of `T` type. If no such variable is found, `defaultValue` is retrieved.

```
public T? Get<T>(string name, T? defaultValue = default)
```

Parameters

`name` string

Name of the variable.

`defaultValue` T

Value, that will be retrieved when no variable with given `name` of `T` type was found.

Returns

T

Variable value or `defaultValue` if no variable with given `name` of `T` type was found.

Type Parameters

`T`

Type of the variable.

Remove(string)

```
public bool Remove(string variableName)
```

Parameters

`variableName` string

Returns

bool

RemoveVariablesWithTag(string)

Attempts to remove all variables with given `tag`. If removal of any of them fails, `false` is retrieved.

```
public bool RemoveVariablesWithTag(string tag)
```

Parameters

`tag` string

Tag by which variables are going to be deleted.

Returns

bool

Whether removal of all variables tagged with `tag` was successfull. If at least one removal failed, `false` is returned.

Set<T>(string, T?, params string[])

```
public void Set<T>(string variableName, T? value, params string[] tags)
```

Parameters

`variableName` string

`value` T

`tags` string[]

Type Parameters

`T`

Namespace TeaPie.Xml

Classes

JUnitXmlWriter

Class JUnitXmlWriter

Namespace: TeaPie.Xml

Assembly: TeaPie.dll

```
public class JUnitXmlWriter : IDisposable
```

Inheritance

object ← JUnitXmlWriter

Implements

IDisposable

Inherited Members

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

Extension Methods

JsonExtensions.ToJsonString(object)

Constructors

JUnitXmlWriter(string)

```
public JUnitXmlWriter(string filePath)
```

Parameters

`filePath` string

Methods

Dispose()

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

```
public void Dispose()
```

EndTestSuite()

```
public void EndTestSuite()
```

EndTestSuitesRoot()

```
public void EndTestSuitesRoot()
```

WriteTestCase(string, string, double, bool, string?, string, string?)

```
public void WriteTestCase(string className, string testName, double timeMs, bool skipped,
string? failureMessage = null, string failureType = "AssertionError", string? stackTrace
= null)
```

Parameters

`className` string

`testName` string

`timeMs` double

`skipped` bool

`failureMessage` string

`failureType` string

`stackTrace` string

WriteTestSuite(string, int, int, int, double, DateTime?)

```
public void WriteTestSuite(string name, int totalTests = 0, int skipped = 0, int failures =
0, double timeMs = 0, DateTime? timestamp = null)
```

Parameters

`name` string

`totalTests` int

`skipped` int

`failures` int

`timeMs` double

`timestamp` DateTime?

WriteTestSuitesRoot(string, int, int, int, double, DateTime?)

```
public void WriteTestSuitesRoot(string name = "", int tests = 0, int skipped = 0, int
failures = 0, double timeMs = 0, DateTime? timestamp = null)
```

Parameters

`name` string

`tests` int

`skipped` int

`failures` int

`timeMs` double

`timestamp` DateTime?