PommaLabs.Thrower

2.2.6

Generated by Doxygen 1.8.11

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 PommaLabs Namespace Reference

Namespaces

- namespace Thrower

5.2 PommaLabs.Thrower Namespace Reference

Classes

- class HttpException

  Represents an exception which contains an error message that should be delivered through the HTTP response, using given status code.
- struct HttpExceptionInfo

  Additional info which will be included into HttpException.
- class Raise

  New exception handling mechanism, which is more fluent than the old ones.
- class RaiseBase

  Stores items shared by various Raise<TEx> instances.
- class ThrowerException

  Exception thrown by Raise<TEx> when the type parameter passed to that class has something invalid (missing constructors, etc).

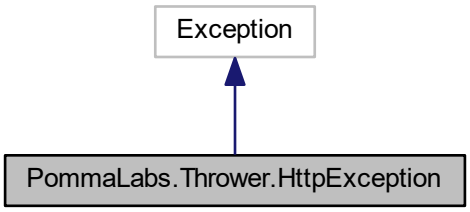
Chapter 6

Class Documentation

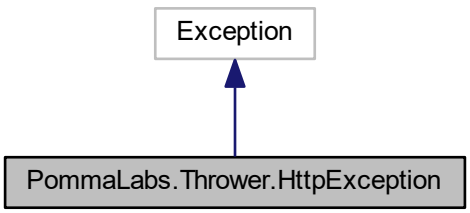
6.1 PommaLabs.Thrower.HttpException Class Reference

Represents an exception which contains an error message that should be delivered through the HTTP response, using given status code.

Inheritance diagram for PommaLabs.Thrower.HttpException:



Collaboration diagram for PommaLabs.Thrower.HttpException:



Public Member Functions

- HttpException (HttpStatusCode httpStatusCode)

  Builds the exception using given status code.
- HttpException (HttpStatusCode httpStatusCode, HttpExceptionInfo additionalInfo)

  Builds the exception using given status code.
- HttpException (HttpStatusCode httpStatusCode, string message)

  Builds the exception using given status code and message.
- HttpException (HttpStatusCode httpStatusCode, string message, HttpExceptionInfo additionalInfo)

  Builds the exception using given status code, message and error code.
- HttpException (HttpStatusCode httpStatusCode, string message, Exception innerException)

  Builds the exception using given status code, message and inner exception.
- HttpException (HttpStatusCode httpStatusCode, string message, Exception innerException, HttpException↩ Info additionalInfo)

  Builds the exception using given status code, message, error code and inner exception.

Properties

- HttpStatusCode HttpStatusCode `[get]`

  The HTTP status code assigned to this exception.
- object ErrorCode `[get]`

  The application defined error code.
- static object DefaultErrorCode `[get, set]`

  The default application defined error code, used when none has been specified.
- string UserMessage = "unspecified" `[get]`

  An error message which can be shown to the user.
- static string DefaultUserMessage `[get, set]`

  The default user message.

6.1.1 Detailed Description

Represents an exception which contains an error message that should be delivered through the HTTP response, using given status code.

Definition at line 70 of file HttpException.cs.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 PommaLabs.Thrower.HttpException.HttpException (HttpStatusCode *httpStatusCode*)

Builds the exception using given status code.

Parameters

httpStatusCode	The HTTP status code.

Definition at line 76 of file HttpException.cs.

6.1.2.2 PommaLabs.Thrower.HttpException.HttpException (HttpStatusCode *httpStatusCode*, HttpExceptionInfo *additionalInfo*)

Builds the exception using given status code.

Parameters

httpStatusCode	The HTTP status code.
additionalInfo	Additional exception info.

Definition at line 86 of file HttpException.cs.

6.1.2.3 PommaLabs.Thrower.HttpException.HttpException (HttpStatusCode *httpStatusCode*, string *message*)

Builds the exception using given status code and message.

Parameters

httpStatusCode	The HTTP status code.
message	The exception message.

Definition at line 100 of file HttpException.cs.

6.1.2.4 PommaLabs.Thrower.HttpException.HttpException (HttpStatusCode *httpStatusCode*, string *message*, HttpExceptionInfo *additionalInfo*)

Builds the exception using given status code, message and error code.

Parameters

httpStatusCode	The HTTP status code.
message	The exception message.
additionalInfo	Additional exception info.

Definition at line 111 of file HttpException.cs.

6.1.2.5 PommaLabs.Thrower.HttpException.HttpException (HttpStatusCode *httpStatusCode*, string *message*, Exception *innerException*)

Builds the exception using given status code, message and inner exception.

Parameters

httpStatusCode	The HTTP status code.
message	The exception message.
innerException	The inner exception.

Definition at line 127 of file HttpException.cs.

6.1.2.6 PommaLabs.Thrower.HttpException.HttpException (HttpStatusCode *httpStatusCode,* string *message,* Exception *innerException,* HttpExceptionInfo *additionalInfo*)

Builds the exception using given status code, message, error code and inner exception.

Parameters

httpStatusCode	The HTTP status code.
message	The exception message.
innerException	The inner exception.
additionalInfo	Additional exception info.

Definition at line 139 of file HttpException.cs.

6.1.3 Property Documentation

6.1.3.1 object PommaLabs.Thrower.HttpException.DefaultErrorCode [static],[get],[set]

The default application defined error code, used when none has been specified.

Definition at line 162 of file HttpException.cs.

6.1.3.2 string PommaLabs.Thrower.HttpException.DefaultUserMessage [static],[get],[set]

The default user message.

Definition at line 172 of file HttpException.cs.

6.1.3.3 object PommaLabs.Thrower.HttpException.ErrorCode [get]

The application defined error code.

Definition at line 157 of file HttpException.cs.

6.1.3.4 HttpStatusCode PommaLabs.Thrower.HttpException.HttpStatusCode [get]

The HTTP status code assigned to this exception.

Definition at line 152 of file HttpException.cs.

6.1.3.5 string PommaLabs.Thrower.HttpException.UserMessage = "unspecified" [get]

An error message which can be shown to the user.

Definition at line 167 of file HttpException.cs.

The documentation for this class was generated from the following file:

- HttpException.cs

6.2 PommaLabs.Thrower.HttpExceptionInfo Struct Reference

Additional info which will be included into HttpException.

Public Member Functions

- HttpExceptionInfo (object errorCode=null, string userMessage=null)
  Builds the additional exception info.

Properties

- object ErrorCode [get, set]
  The application defined error code.
- string UserMessage [get, set]
  An error message which can be shown to user.

6.2.1 Detailed Description

Additional info which will be included into HttpException.

Definition at line 38 of file HttpException.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 PommaLabs.Thrower.HttpExceptionInfo.HttpExceptionInfo (object *errorCode* = null, string *userMessage* = null)

Builds the additional exception info.

Parameters

errorCode	The application defined error code.
userMessage	The user message.

Definition at line 45 of file HttpException.cs.

6.2.3 Property Documentation

6.2.3.1 object PommaLabs.Thrower.HttpExceptionInfo.ErrorCode [get], [set]

The application defined error code.

Definition at line 55 of file HttpException.cs.

6.2.3.2 string PommaLabs.Thrower.HttpExceptionInfo.UserMessage [get], [set]

An error message which can be shown to user.

Definition at line 61 of file HttpException.cs.

The documentation for this struct was generated from the following file:

- HttpException.cs

6.3 PommaLabs.Thrower.Raise Class Reference

New exception handling mechanism, which is more fluent than the old ones.

6.3.1 Detailed Description

New exception handling mechanism, which is more fluent than the old ones.

Definition at line 32 of file Raise.cs.

The documentation for this class was generated from the following file:

- Raise.cs

6.4 PommaLabs.Thrower.Raise Class Reference

New exception handling mechanism, which is more fluent than the old ones.

6.4.1 Detailed Description

New exception handling mechanism, which is more fluent than the old ones.

Definition at line 32 of file Raise.cs.

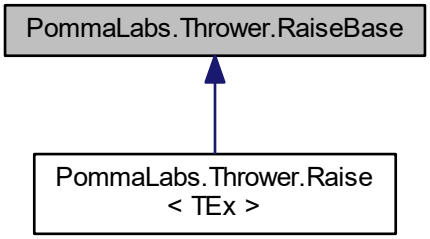
The documentation for this class was generated from the following file:

- Raise.cs

6.5 PommaLabs.Thrower.RaiseBase Class Reference

Stores items shared by various Raise<TEx> instances.

Inheritance diagram for PommaLabs.Thrower.RaiseBase:



Static Protected Attributes

- static readonly Type[ ] NoCtorTypes = new Type[0]

  Stores an empty array of System.Type used to seek constructors without parameters.
- static readonly Type[ ] StrExCtorTypes = { typeof(string), typeof(Exception) }

  Stores the types needed to seek the constructor which takes a string and an exception as parameters to instance the exception.
- static readonly Type[ ] StrCtorType = { typeof(string) }

  Stores the type needed to seek the constructor which takes a string as parameter to instance the exception.

6.5.1 Detailed Description

Stores items shared by various Raise<TEx> instances.

Definition at line 36 of file RaiseGeneric.cs.

6.5.2 Member Data Documentation

6.5.2.1 readonly Type [ ] PommaLabs.Thrower.RaiseBase.NoCtorTypes = new Type[0] [static], [protected]

Stores an empty array of System.Type used to seek constructors without parameters.

Definition at line 43 of file RaiseGeneric.cs.

6.5.2.2 readonly Type [ ] PommaLabs.Thrower.RaiseBase.StrCtorType = { typeof(string) } [static], [protected]

Stores the type needed to seek the constructor which takes a string as parameter to instance the exception.

Definition at line 59 of file RaiseGeneric.cs.

6.5.2.3 readonly Type [ ] PommaLabs.Thrower.RaiseBase.StrExCtorTypes = { typeof(string), typeof(Exception) } [static], [protected]

Stores the types needed to seek the constructor which takes a string and an exception as parameters to instance the exception.

Definition at line 51 of file RaiseGeneric.cs.

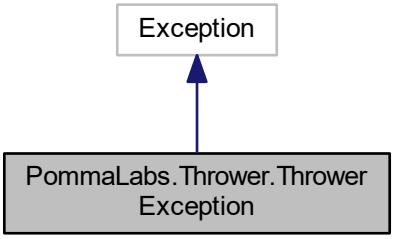
The documentation for this class was generated from the following file:

- RaiseGeneric.cs

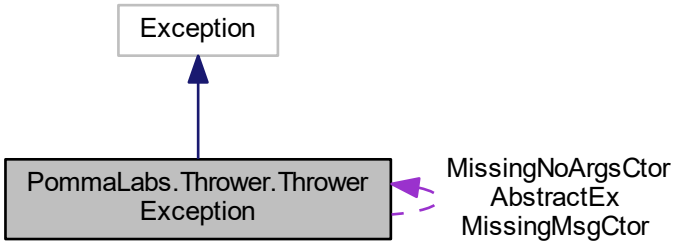
6.6 PommaLabs.Thrower.ThrowerException Class Reference

Exception thrown by Raise<TEx> when the type parameter passed to that class has something invalid (missing constructors, etc).

Inheritance diagram for PommaLabs.Thrower.ThrowerException:



Collaboration diagram for PommaLabs.Thrower.ThrowerException:



6.6.1 Detailed Description

Exception thrown by Raise<TEx> when the type parameter passed to that class has something invalid (missing constructors, etc).

Definition at line 35 of file ThrowerException.cs.

The documentation for this class was generated from the following file:

- ThrowerException.cs

Chapter 7

File Documentation

7.1 HttpException.cs File Reference

Classes

- struct PommaLabs.Thrower.HttpExceptionInfo

  Additional info which will be included into HttpException.
- class PommaLabs.Thrower.HttpException

  Represents an exception which contains an error message that should be delivered through the HTTP response, using given status code.

Namespaces

- namespace PommaLabs.Thrower

7.2 HttpException.cs

```
00001 // File name: HttpException.cs
00002 //
00003 // Author(s): Alessio Parma <alessio.parma@gmail.com>
00004 //
00005 // The MIT License (MIT)
00006 //
00007 // Copyright (c) 2013-2016 Alessio Parma <alessio.parma@gmail.com>
00008 //
00009 // Permission is hereby granted, free of charge, to any person obtaining a copy of this software and
00010 // associated documentation files (the "Software"), to deal in the Software without restriction,
00011 // including without limitation the rights to use, copy, modify, merge, publish, distribute,
00012 // sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is
00013 // furnished to do so, subject to the following conditions:
00014 //
00015 // The above copyright notice and this permission notice shall be included in all copies or
00016 // substantial portions of the Software.
00017 //
00018 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
00019 // NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
00020 // NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
00021 // DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT
00022 // OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00023
00024 using PommaLabs.Thrower.Validation;
00025 using System;
00026 using System.Collections;
00027 using System.Collections.Generic;
00028 using System.Diagnostics.CodeAnalysis;
```

```
00029 using System.Net;
00030 using System.Runtime.Serialization;
00031
00032 namespace PommaLabs.Thrower
00033 {
00037     [Serializable]
00038     public struct HttpExceptionInfo
00039     {
00045         public HttpExceptionInfo(object errorCode = null, string userMessage = null)
00046         {
00047             ErrorCode = errorCode ?? HttpException.DefaultErrorCode;
00048             UserMessage = userMessage ?? HttpException.
      DefaultUserMessage;
00049         }
00050
00054         [Validate(Required = false)]
00055         public object ErrorCode { get; set; }
00056
00060         [Validate(Required = false)]
00061         public string UserMessage { get; set; }
00062     }
00063
00068     [Serializable]
00069     [SuppressMessage("Microsoft.Design", "CA1032:ImplementStandardExceptionConstructors")]
00070     public sealed class HttpException : Exception
00071     {
00076         public HttpException(HttpStatusCode httpStatusCode)
00077             : this(httpStatusCode, new HttpExceptionInfo())
00078         {
00079         }
00080
00086         public HttpException(HttpStatusCode httpStatusCode,
      HttpExceptionInfo additionalInfo)
00087         {
00088             HttpStatusCode = httpStatusCode;
00089             ErrorCode = additionalInfo.ErrorCode ?? DefaultErrorCode;
00090             UserMessage = additionalInfo.UserMessage ?? DefaultUserMessage;
00091
00092             CustomizeException();
00093         }
00094
00100         public HttpException(HttpStatusCode httpStatusCode, string message)
00101             : this(httpStatusCode, message, new HttpExceptionInfo())
00102         {
00103         }
00104
00111         public HttpException(HttpStatusCode httpStatusCode, string message,
      HttpExceptionInfo additionalInfo)
00112             : base(message)
00113         {
00114             HttpStatusCode = httpStatusCode;
00115             ErrorCode = additionalInfo.ErrorCode ?? DefaultErrorCode;
00116             UserMessage = additionalInfo.UserMessage ?? DefaultUserMessage;
00117
00118             CustomizeException();
00119         }
00120
00127         public HttpException(HttpStatusCode httpStatusCode, string message, Exception
      innerException)
00128             : this(httpStatusCode, message, innerException, new
      HttpExceptionInfo())
00129         {
00130         }
00131
00139         public HttpException(HttpStatusCode httpStatusCode, string message, Exception
      innerException, HttpExceptionInfo additionalInfo)
00140             : base(message, innerException)
00141         {
00142             HttpStatusCode = httpStatusCode;
00143             ErrorCode = additionalInfo.ErrorCode ?? DefaultErrorCode;
00144             UserMessage = additionalInfo.UserMessage ?? DefaultUserMessage;
00145
00146             CustomizeException();
00147         }
00148
00152         public HttpStatusCode HttpStatusCode { get; }
00153
00157         public object ErrorCode { get; }
00158
00162         public static object DefaultErrorCode { get; set; } = "unspecified";
00163
00167         public string UserMessage { get; }
00168
00172         public static string DefaultUserMessage { get; set; } = "unspecified";
00173
00174         private void CustomizeException()
00175         {
```

```
00176             HResult = (int) HttpStatusCode;
00177
00178             Data.Add(nameof(HttpStatusCode), HttpStatusCode.ToString());
00179             Data.Add(nameof(ErrorCode), ErrorCode?.ToString());
00180             Data.Add(nameof(UserMessage), UserMessage);
00181         }
00182     }
00183 }
```

7.3 Raise.cs File Reference

Classes

- class PommaLabs.Thrower.Raise

  New exception handling mechanism, which is more fluent than the old ones.

Namespaces

- namespace PommaLabs.Thrower

7.4 Raise.cs

```
00001 // File name: Raise.cs
00002 //
00003 // Author(s): Alessio Parma <alessio.parma@gmail.com>
00004 //
00005 // The MIT License (MIT)
00006 //
00007 // Copyright (c) 2013-2016 Alessio Parma <alessio.parma@gmail.com>
00008 //
00009 // Permission is hereby granted, free of charge, to any person obtaining a copy of this software and
00010 // associated documentation files (the "Software"), to deal in the Software without restriction,
00011 // including without limitation the rights to use, copy, modify, merge, publish, distribute,
00012 // sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is
00013 // furnished to do so, subject to the following conditions:
00014 //
00015 // The above copyright notice and this permission notice shall be included in all copies or
00016 // substantial portions of the Software.
00017 //
00018 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
00019 // NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
00020 // NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
00021 // DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT
00022 // OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00023
00024 using PommaLabs.Thrower.ExceptionHandlers;
00025 using System.Diagnostics.CodeAnalysis;
00026
00027 namespace PommaLabs.Thrower
00028 {
00032     public static class Raise
00033     {
00037         public static ArgumentExceptionHandler ArgumentException { get; } = new ArgumentExceptionHandler();
00038
00042         public static ArgumentNullExceptionHandler ArgumentNullException { get; } = new
      ArgumentNullExceptionHandler();
00043
00047         public static ArgumentOutOfRangeExceptionHandler ArgumentOutOfRangeException { get; } = new
      ArgumentOutOfRangeExceptionHandler();
00048
00052         public static HttpExceptionHandler HttpException { get; } = new HttpExceptionHandler()
      ;
00053
00057         public static IndexOutOfRangeExceptionHandler IndexOutOfRangeException { get; } = new
      IndexOutOfRangeExceptionHandler();
00058
00062         public static InvalidOperationExceptionHandler InvalidOperationException { get; } = new
      InvalidOperationExceptionHandler();
00063
00067         public static NotSupportedExceptionHandler NotSupportedException { get; } = new
      NotSupportedExceptionHandler();
00068
00072         public static ObjectDisposedExceptionHandler ObjectDisposedException { get; } = new
      ObjectDisposedExceptionHandler();
00073     }
00074 }
```

7.5 RaiseGeneric.cs File Reference

Classes

- class PommaLabs.Thrower.RaiseBase

  Stores items shared by various Raise<TEx> instances.
- class PommaLabs.Thrower.Raise< TEx >

  Contains methods that throw specified exception TEx if given conditions will be verified.

Namespaces

- namespace PommaLabs.Thrower

7.6 RaiseGeneric.cs

```
00001 // File name: RaiseGeneric.cs
00002 //
00003 // Author(s): Alessio Parma <alessio.parma@gmail.com>
00004 //
00005 // The MIT License (MIT)
00006 //
00007 // Copyright (c) 2013-2016 Alessio Parma <alessio.parma@gmail.com>
00008 //
00009 // Permission is hereby granted, free of charge, to any person obtaining a copy of this software and
00010 // associated documentation files (the "Software"), to deal in the Software without restriction,
00011 // including without limitation the rights to use, copy, modify, merge, publish, distribute,
00012 // sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is
00013 // furnished to do so, subject to the following conditions:
00014 //
00015 // The above copyright notice and this permission notice shall be included in all copies or
00016 // substantial portions of the Software.
00017 //
00018 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
00019 // NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
00020 // NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
00021 // DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT
00022 // OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00023
00024 using PommaLabs.Thrower.Reflection;
00025 using System;
00026 using System.Collections.Generic;
00027 using System.Diagnostics.CodeAnalysis;
00028 using System.Linq;
00029 using System.Reflection;
00030
00031 namespace PommaLabs.Thrower
00032 {
00036     public abstract class RaiseBase
00037     {
00041         [SuppressMessage("Microsoft.Naming", "CA1704:IdentifiersShouldBeSpelledCorrectly")]
00042         [SuppressMessage("Microsoft.Security", "CA2105:ArrayFieldsShouldNotBeReadOnly")]
00043         protected static readonly Type[] NoCtorTypes = new Type[0];
00044
00049         [SuppressMessage("Microsoft.Naming", "CA1704:IdentifiersShouldBeSpelledCorrectly")]
00050         [SuppressMessage("Microsoft.Security", "CA2105:ArrayFieldsShouldNotBeReadOnly")]
00051         protected static readonly Type[] StrExCtorTypes = { typeof(string), typeof(Exception) };
00052
00057         [SuppressMessage("Microsoft.Naming", "CA1704:IdentifiersShouldBeSpelledCorrectly")]
00058         [SuppressMessage("Microsoft.Security", "CA2105:ArrayFieldsShouldNotBeReadOnly")]
00059         protected static readonly Type[] StrCtorType = { typeof(string) };
00060     }
00061
00071     public sealed partial class Raise<TEx> : RaiseBase where TEx : Exception
00072     {
00073 #pragma warning disable RECS0108 // Warns about static fields in generic types
00074
00079         private static readonly bool ExTypeIsAbstract = PortableTypeInfo.IsAbstract(typeof(TEx));
00080
00085         private static readonly ConstructorInfo NoArgsCtor = GetCtor(NoCtorTypes);
00086
00099         private static readonly ConstructorInfo MsgCtor = GetCtor(StrExCtorTypes) ?? GetCtor(StrCtorType);
00100
```

```
00105         private static readonly int MsgArgCount = (MsgCtor == null) ? 0 : MsgCtor.GetParameters().Length;
00106
00107 #pragma warning restore RECS0108 // Warns about static fields in generic types
00108
00112         private Raise()
00113         {
00114             throw new InvalidOperationException("This class should not be instantiated");
00115         }
00116
00131         [SuppressMessage("Microsoft.Design", "CA1000:DoNotDeclareStaticMembersOnGenericTypes")]
00132         public static void If(bool cond)
00133         {
00134             if (cond)
00135             {
00136                 DoThrow();
00137             }
00138         }
00139
00161         [SuppressMessage("Microsoft.Design", "CA1000:DoNotDeclareStaticMembersOnGenericTypes")]
00162         public static void If(bool cond, string message)
00163         {
00164             if (cond)
00165             {
00166                 DoThrow(message);
00167             }
00168         }
00169
00184         [SuppressMessage("Microsoft.Design", "CA1000:DoNotDeclareStaticMembersOnGenericTypes")]
00185         public static void IfNot(bool cond)
00186         {
00187             if (!cond)
00188             {
00189                 DoThrow();
00190             }
00191         }
00192
00214         [SuppressMessage("Microsoft.Design", "CA1000:DoNotDeclareStaticMembersOnGenericTypes")]
00215         public static void IfNot(bool cond, string message)
00216         {
00217             if (!cond)
00218             {
00219                 DoThrow(message);
00220             }
00221         }
00222
00223         private static ConstructorInfo GetCtor(System.Collections.Generic.IList<Type> ctorTypes)
00224         {
00225             return (from c in PortableTypeInfo.GetConstructors(typeof(TEx))
00226                     let args = c.GetParameters()
00227                     let zipArgs = MyZip(args, ctorTypes, (argType, ctorType) => new { argType, ctorType })
00228                     where args.Length == ctorTypes.Count &&
00229                           (c.IsPublic || c.IsAssembly) &&
00230                           zipArgs.All(t => ReferenceEquals(t.argType.ParameterType, t.ctorType))
00231                     select c).FirstOrDefault();
00232         }
00233
00234         private static IEnumerable<TResult> MyZip<TFirst, TSecond, TResult>(IEnumerable<TFirst> first,
      IEnumerable<TSecond> second, Func<TFirst, TSecond, TResult> resultSelector)
00235         {
00236             Raise.ArgumentNullException.IfIsNull(first, nameof(first));
00237             Raise.ArgumentNullException.IfIsNull(second, nameof(second));
00238             Raise.ArgumentNullException.IfIsNull(resultSelector, nameof(resultSelector));
00239
00240             using (IEnumerator<TFirst> e1 = first.GetEnumerator())
00241             using (IEnumerator<TSecond> e2 = second.GetEnumerator())
00242             {
00243                 while (e1.MoveNext() && e2.MoveNext())
00244                 {
00245 #pragma warning disable CC0031 // Check for null before calling a delegate
00246                     yield return resultSelector(e1.Current, e2.Current);
00247 #pragma warning restore CC0031 // Check for null before calling a delegate
00248                 }
00249             }
00250         }
00251
00252         private static void DoThrow()
00253         {
00254             // Checks whether the proper constructor exists. If not, then we produce an internal exception.
00255             if (ExTypeIsAbstract)
00256             {
00257                 throw ThrowerException.AbstractEx;
00258             }
00259             if (NoArgsCtor == null)
00260             {
00261                 throw ThrowerException.MissingNoArgsCtor;
00262             }
00263             // A proper constrctor exists: therefore, we can throw the exception.
```

```
00264             throw (TEx) NoArgsCtor.Invoke(new object[0]);
00265         }
00266 
00267         private static void DoThrow(string message)
00268         {
00269             // Checks whether the proper constructor exists. If not, then we produce an internal exception.
00270             if (ExTypeIsAbstract)
00271             {
00272                 throw ThrowerException.AbstractEx;
00273             }
00274             if (MsgCtor == null)
00275             {
00276                 throw ExTypeIsAbstract ? ThrowerException.AbstractEx :
      ThrowerException.MissingMsgCtor;
00277             }
00278             // A proper constrctor exists: therefore, we can throw the exception.
00279             var messageArgs = new object[MsgArgCount];
00280             messageArgs[0] = message;
00281             throw (TEx) MsgCtor.Invoke(messageArgs);
00282         }
00283     }
00284 }
```

7.7 ThrowerException.cs File Reference

Classes

- class PommaLabs.Thrower.ThrowerException

  Exception thrown by Raise<TEx> when the type parameter passed to that class has something invalid (missing constructors, etc).

Namespaces

- namespace PommaLabs.Thrower

7.8 ThrowerException.cs

```
00001 // File name: ThrowerException.cs
00002 //
00003 // Author(s): Alessio Parma <alessio.parma@gmail.com>
00004 //
00005 // The MIT License (MIT)
00006 //
00007 // Copyright (c) 2013-2016 Alessio Parma <alessio.parma@gmail.com>
00008 //
00009 // Permission is hereby granted, free of charge, to any person obtaining a copy of this software and
00010 // associated documentation files (the "Software"), to deal in the Software without restriction,
00011 // including without limitation the rights to use, copy, modify, merge, publish, distribute,
00012 // sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is
00013 // furnished to do so, subject to the following conditions:
00014 //
00015 // The above copyright notice and this permission notice shall be included in all copies or
00016 // substantial portions of the Software.
00017 //
00018 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
00019 // NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
00020 // NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
00021 // DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT
00022 // OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
00023 
00024 using System;
00025 using System.Diagnostics.CodeAnalysis;
00026 
00027 namespace PommaLabs.Thrower
00028 {
00033     [Serializable]
00034     [SuppressMessage("Microsoft.Design", "CA1032:ImplementStandardExceptionConstructors")]
00035     public sealed class ThrowerException : Exception
00036     {
```

```
00037         [SuppressMessage("Microsoft.Design", "CA1032:ImplementStandardExceptionConstructors")]
00038         private ThrowerException(string message)
00039             : base(message)
00040         {
00041         }
00042 
00043         internal static ThrowerException AbstractEx => new ThrowerException("Given exception type is
      abstract");
00044 
00045         internal static ThrowerException MissingNoArgsCtor => new ThrowerException("Given exception type
      has no parameterless constructor");
00046 
00047         internal static ThrowerException MissingMsgCtor => new ThrowerException("Given exception type has
      not a valid message constructor");
00048     }
00049 }
```

Index