

Исключения и обработка ошибок

Юрий Литвинов
yurii.litvinov@gmail.com

15.03.2019г

Бросание исключений

```
if (t == null)
{
    throw new NullPointerException();
}

throw new NullPointerException("Hello!");
```

Обработка исключений

```
try {  
    // Код, который может бросать исключения  
} catch (Type1 id1) {  
    // Обработка исключений типа Type1  
} catch (Type2 id2) {  
    // Обработка исключений типа Type2  
} catch {  
    // Обработка всех оставшихся исключений  
} finally {  
    // Код, выполняющийся в любом случае  
}
```

Что делается

- ▶ На куче создаётся объект-исключение
- ▶ Исполнение метода прерывается
- ▶ Если в вызывающем методе есть обработчик этого исключения, вызывается он, иначе исполнение вызывающего тоже прерывается
- ▶ И т.д., пока не найдём обработчик или прервём Main

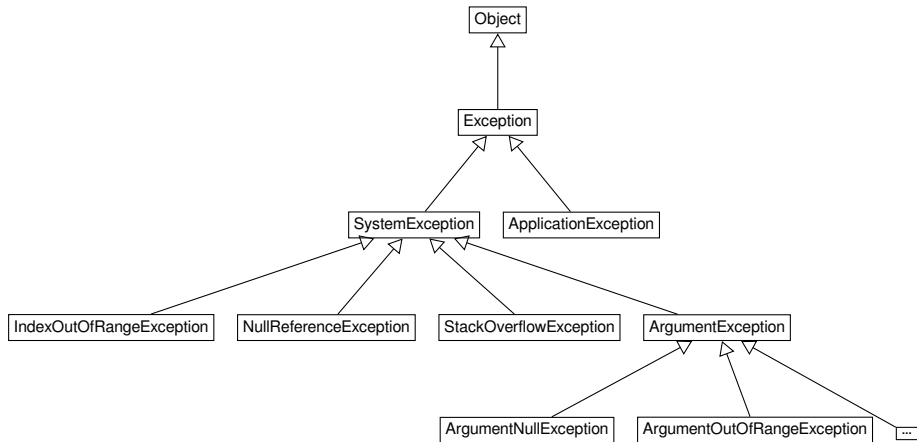
Пример

```
private void ReadData(String pathname)
{
    FileStream fs = null;
    try {
        fs = new FileStream(pathname, FileMode.Open);
        // Делать что-то полезное
    }
    catch (IOException) {
        // Код восстановления после ошибки
    }
    finally {
        // Закрывать файл надо в любом случае
        if (fs != null) fs.Close();
    }
}
```

Фильтры исключений

```
public static string MakeRequest()
{
    try
    {
        var request = WebRequest.Create("http://se.math.spbu.ru");
        var response = request.GetResponse();
        var stream = response.GetResponseStream();
        var reader = new StreamReader(stream);
        var data = reader.ReadToEnd();
        return data;
    }
    catch (WebException e)
        when (e.Status == WebExceptionStatus.ConnectFailure)
    {
        return "Connection failed";
    }
}
```

Иерархия классов-исключений



Свойства класса Exception

- ▶ Data
- ▶ HelpLink
- ▶ InnerException
- ▶ Message
- ▶ Source
- ▶ StackTrace
- ▶ HResult

Пример (плохой)

```
try
{
    throw new Exception("Something is very wrong");
}
catch (Exception e)
{
    Console.WriteLine("Caught Exception");
    Console.WriteLine("e.Message: " + e.Message);
    Console.WriteLine("e.ToString(): " + e.ToString());
    Console.WriteLine("e.StackTrace:\n" + e.StackTrace);
}
```

Что выведется

Caught Exception

e.Message: Something **is** very wrong

e.ToString(): System.Exception: Something **is** very wrong

в CSharpConsoleApplication.Program.Main(String[] args) в

c:\Projects\CSharpConsoleApplication\CSharpConsoleApplication\Program.cs: строка 15

e.StackTrace:

в CSharpConsoleApplication.Program.Main(String[] args) в

c:\Projects\CSharpConsoleApplication\CSharpConsoleApplication\Program.cs: строка 15

Для продолжения нажмите любую клавишу . . .

Перебрасывание исключений

```
try
{
    throw new Exception("Something is wrong");
}
catch (Exception e)
{
    Console.WriteLine("Caught Exception");
    throw; // "throw e;" тоже работает, но сбросит stack trace
}
```

Или

```
throw new Exception("Outer exception", e);
```

Свои классы-исключения

```
public class MyException : Exception
{
    public MyException()
    {
    }

    public MyException(string message)
        : base(message)
    {
    }
}
```

Идеологически правильное объявление исключения

[Serializable]

```
public class MyException : Exception
{
    public MyException() { }
    public MyException(string message) : base(message) { }
    public MyException(string message, Exception inner)
        : base(message, inner) { }
    protected MyException(
        System.Runtime.Serialization.SerializationInfo info,
        System.Runtime.Serialization.StreamingContext context)
        : base(info, context) { }
}
```

“Интересные” классы-исключения

- ▶ Corrupted state exceptions (CSE) — не ловятся catch-ем
 - ▶ StackOverflowException
 - ▶ AccessViolationException
 - ▶ System.Runtime.InteropServices.SEHException
- ▶ FileLoadException, BadImageFormatException, InvalidProgramException, ...
- ▶ ThreadAbortException
- ▶ TypeInitializationException
- ▶ TargetInvocationException
- ▶ OutOfMemoryException
- ▶ Можно кидать null. При этом рантайм сам кидает NullReferenceException.

Environment.FailFast

```
try
{
    // Делать что-то полезное
}
catch (OutOfMemoryException e)
{
    Console.WriteLine("Закончилась память :(");
    Environment.FailFast(
        String.Format($"Out of Memory: {e.Message}"));
}
```

Исключения и тесты

```
[TestMethod]
[ExpectedException(typeof(System.DivideByZeroException))]
public void DivideTest()
{
    DivisionClass target = new DivisionClass();
    int numerator = 4;
    int denominator = 0;
    int actual = target.Divide(numerator, denominator);
}
```


Исключения, best practices

- ▶ Не бросать исключения без нужды
 - ▶ В нормальном режиме работы исключения бросаться не должны вообще
- ▶ Свои исключения наследовать от `System.Exception`
- ▶ Документировать все свои исключения, бросаемые методом
- ▶ Не ловить `Exception`, `SystemException`
 - ▶ Исключения, указывающие на ошибку в коде (например, `NullReferenceException`) уж точно не ловить
- ▶ По возможности кидать библиотечные исключения или их наследников:
 - ▶ `InvalidOperationException`
 - ▶ `ArgumentException`
- ▶ Имя класса должно заканчиваться на “Exception”
- ▶ Код должен быть безопасным с точки зрения исключений
 - ▶ Не забывать про `finally`
 - ▶ Или `using`, `lock`, `foreach`, которые тоже генерят `finally`

using, IDisposable

```
public static class Program {  
    public static void Main() {  
        // Create the bytes to write to the temporary file.  
        Byte[] bytesToWrite = new Byte[] { 1, 2, 3, 4, 5 };  
        // Create the temporary file.  
        using (FileStream fs = new FileStream("Temp.dat", FileMode.Create)) {  
            // Write the bytes to the temporary file.  
            fs.Write(bytesToWrite, 0, bytesToWrite.Length);  
        }  
        // Delete the temporary file.  
        File.Delete("Temp.dat");  
    }  
}
```

Особенности современного C#

Кортежи и деконструкция

```
static (int prev, int cur) Fibonacci(int n)
{
    var (prevPrev, prev) = n <= 2 ? (0, 1) : Fibonacci(n - 1);
    return (prev, prevPrev + prev);
}

private static void Main(string[] args)
{
    var (_, result) = Fibonacci(7);
    Console.WriteLine(result);
}
```

Именованные поля кортежей

```
int count = 5;  
string label = "Colors used in the map";  
var pair = (count: count, label: label);
```

```
int count = 5;  
string label = "Colors used in the map";  
var pair = (count, label); // element names are "count" and "label"
```

Инициализация свойств

```
public class Person
{
    public int Age { get; set; } = 0;
    public string Name { get; set; } = "Anonymous";
}
```

Null propagation

```
var first = person?.FirstName;
```

```
int? age = person?.Age;
```

```
if (age.HasValue)
```

```
{
```

```
    int realAge = age.Value;
```

```
}
```

```
var otherFirst = person?.FirstName ?? "Unspecified";
```

Локальные функции

```
public int Fibonacci(int x)
{
    if (x < 0)
        throw new ArgumentException("Less negativity please!", nameof(x));
    return Fib(x).current;

    (int current, int previous) Fib(int i)
    {
        if (i == 0) return (1, 0);
        var (p, pp) = Fib(i - 1);
        return (p + pp, p);
    }
}
```

Индексеры

```
class SampleCollection {  
    private int[] arr = new int[100];  
    public int this[int i]  
    {  
        get  
        {  
            return arr[i];  
        }  
        set  
        {  
            arr[i] = value;  
        }  
    }  
}
```


Индексная инициализация

```
private Dictionary<int, string> webErrors = new Dictionary<int, string>
{
    [404] = "Page not Found",
    [302] = "Page moved, but left a forwarding address.",
    [500] = "The web server can't come out to play today."
};
```

Именованные и необязательные аргументы

```
PrintOrderDetails(productName: "Red Mug", 31, "Gift Shop");
```

```
public void ExampleMethod(int required,  
    string optionalstr = "default string",  
    int optionalint = 10)
```

```
anExample.ExampleMethod(3, optionalint: 4);
```

params

```
public static void UseParams(params int[] list)
{
    for (int i = 0; i < list.Length; i++)
    {
        Console.Write(list[i] + " ");
    }
    Console.WriteLine();
}

...
UseParams(1, 2, 3, 4);
```