

LABORATORIOS CLASE 11

Detallamos los ejercicios de este laboratorio. **Se recomienda realizarlos todos!**. El objetivo es asegurar la ejercitación adecuada para los contenidos de esta clase:

- Ejercicio 1: Excepciones
- Ejercicio 2: Aplicar distintos tipos de excepciones en una aplicación web
- Ejercicio 3: Control de excepciones en el proyecto integrador

Ejercicio 1: Excepciones

Ejemplo 1

El siguiente ejemplo pone de manifiesto el flujo de ejecución que sigue un programa que lanza y procesa una excepción.

1. Cree un nuevo proyecto de consola en una nueva solución. Identifique al proyecto y a la solución con el nombre **Excepciones1**.
2. Analice y copie el siguiente ejemplo básico de manejo de excepciones.

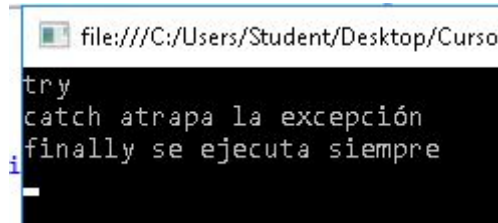
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Excepciones1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Ejemplo 1
            try
            {
                Console.WriteLine("try");
                throw new Exception("Mi excepción");
            }
            catch
            {
                Console.WriteLine("catch atrapa la excepción");
            }
            finally
            {
                Console.WriteLine("finally se ejecuta siempre");
            }
        }
    }
}
```

```

        Console.ReadKey();
    }
}

```



Ejemplo 2

Se puede profundizar en el tratamiento de la excepción, por ejemplo, comprobando alguna propiedad del objeto **Exception** generado.

La clase **Exception** es la clase base de la que derivan las excepciones. La mayoría de los objetos de excepción son instancias de alguna clase derivada de **Exception**, pero se puede iniciar cualquier objeto derivado de la clase **Object** como excepción. En casi todos los casos, es recomendable iniciar y detectar sólo objetos **Exception**.

La clase **Exception** tiene varias propiedades que facilitan la comprensión de una excepción. Entre éstas destacamos la propiedad **Message**. Esta propiedad proporciona información sobre la causa de una excepción. Veamos cómo se utiliza:

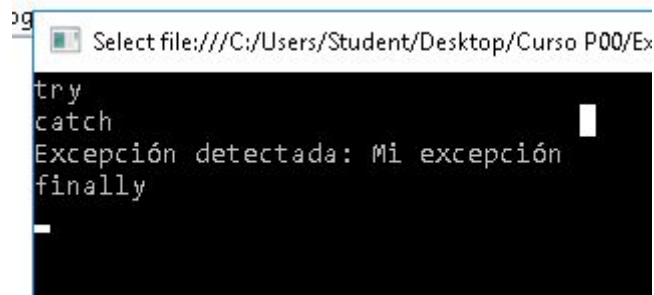
1. Comente el código del ejemplo 1 y agregue a continuación el de este ejemplo 2. Observe que se personaliza la excepción en **el throw new** y en el primer **catch** se puede obtener esa personalización a través de la propiedad **Message**.

```

// Ejemplo 2
try
{
    Console.WriteLine("try");
    throw new Exception("Mi excepción");
}
catch (Exception e)
{
    Console.WriteLine("catch");
    Console.WriteLine("Excepción detectada: " + e.Message);
}
catch
{
    Console.WriteLine("catch");
}
finally
{
    Console.WriteLine("finally");
}

```

```
Console.ReadKey();
```



```
try
catch
Excepción detectada: Mi excepción
finally
```

Ejemplo 3

El siguiente ejemplo muestra cómo el uso de excepciones puede controlar un número importante de situaciones de error.

1. Cree un nuevo proyecto de consola en una nueva solución. Identifique al proyecto y a la solución con el nombre **Excepciones2**.
2. Analice y copie el siguiente ejemplo básico de manejo de excepciones.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Excepciones2
{
    class Program
    {
        static void Main(string[] args)
        {
            int numerador = 10;
            Console.WriteLine("Numerador es = {0}", numerador);
            Console.Write("Denominador = ");
            string strDen = Console.ReadLine();

            int denominador, cociente;

            // Trata de hacer una división
            try
            {
```

```

        Console.WriteLine("--> try");
        denominador = Convert.ToInt16(strDen);
        cociente = numerador / denominador;
        Console.WriteLine("Cociente = {0}", cociente);
    }

    // Atrapa excepción ArithmeticException
    catch (ArithmeticException e)
    {
        Console.WriteLine("--> catch");
        Console.WriteLine("Excepción aritmética");
        Console.WriteLine("ArithmeticException Handler: {0}",
            e.ToString());
    }

    // Atrapa excepción ArgumentNullException
    catch (ArgumentNullException e)
    {
        Console.WriteLine("--> catch");
        Console.WriteLine("Excepción de argumento nulo");
        Console.WriteLine("ArgumentNullException Handler: {0}",
            e.ToString());
    }

    // Atrapa excepción General
    catch (Exception e)
    {
        Console.WriteLine("--> catch");
        Console.WriteLine("generic Handler: {0}",
            e.ToString());
    }
    finally
    {
        Console.WriteLine("--> finally");
    }

    Console.ReadLine();
}
}
}

```

3. Para la primera ejecución, **ingrese como denominador el valor 2**. No producirá ninguna excepción. Observe el resultado.

Ejemplo 4

Hemos visto que pueden conocerse los detalles de la excepción que se haya producido.

Podemos conocer más detalles usando la propiedad **StackTrace**. Esta propiedad contiene un seguimiento de pila que se puede utilizar para determinar dónde se ha producido un error. El seguimiento de pila contiene el nombre del archivo de código fuente y el número de la línea del programa si está disponible la información de depuración.

1. Comente el código del main del ejercicio anterior y copie a continuación este ejemplo previo su análisis. Deberá agregar la librería **System.Diagnostics**.

```
static void Main(string[] args)
{
    int numerador = 10;
    Console.WriteLine("Numerador es = {0}", numerador);

    Console.Write("Denominador = ");
    string strDen = Console.ReadLine();

    int denominador, cociente;

    try
    {
        Console.WriteLine("--> try");
        denominador = Convert.ToInt16(strDen);
        cociente = numerador / denominador;
        Console.WriteLine("Cociente = {0}", cociente);
    }

    catch (Exception e)
    {
        Console.WriteLine("--> catch");
        Console.WriteLine("Generic Handler: {0}", e.ToString());
        Console.WriteLine();

        StackTrace st = new StackTrace(e, true);

        Console.WriteLine("Traza de la pila:");
        for (int i = 0; i < st.FrameCount; i++)
        {
            StackFrame sf = st.GetFrame(i);
            Console.WriteLine("  Pila de llamadas, Método: {0}",
                              sf.GetMethod());
            Console.WriteLine("  Pila de llamadas, Línea : {0}",
                              sf.GetFileLineNumber());
        }
    }
}
```

```

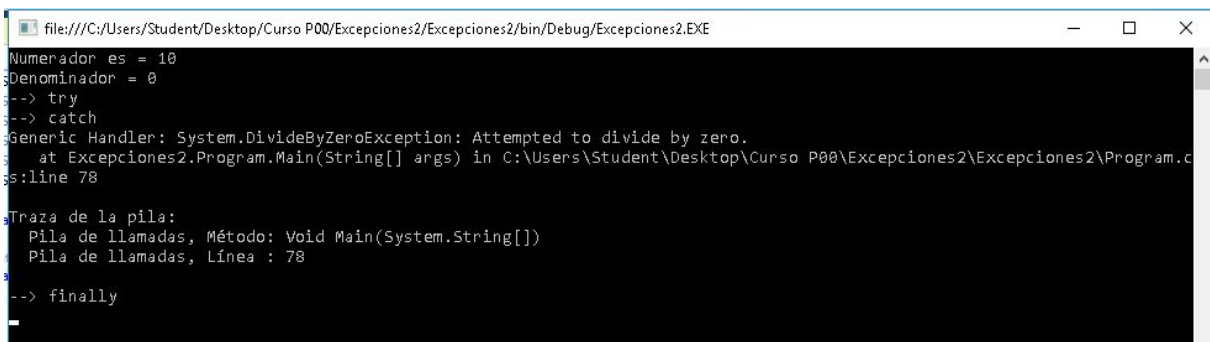
        Console.WriteLine();
    }

    finally
    {
        Console.WriteLine("--> finally");
    }

    Console.ReadLine();
}

```

2. Pruebe la ejecución **ingresando un cero como denominador**.



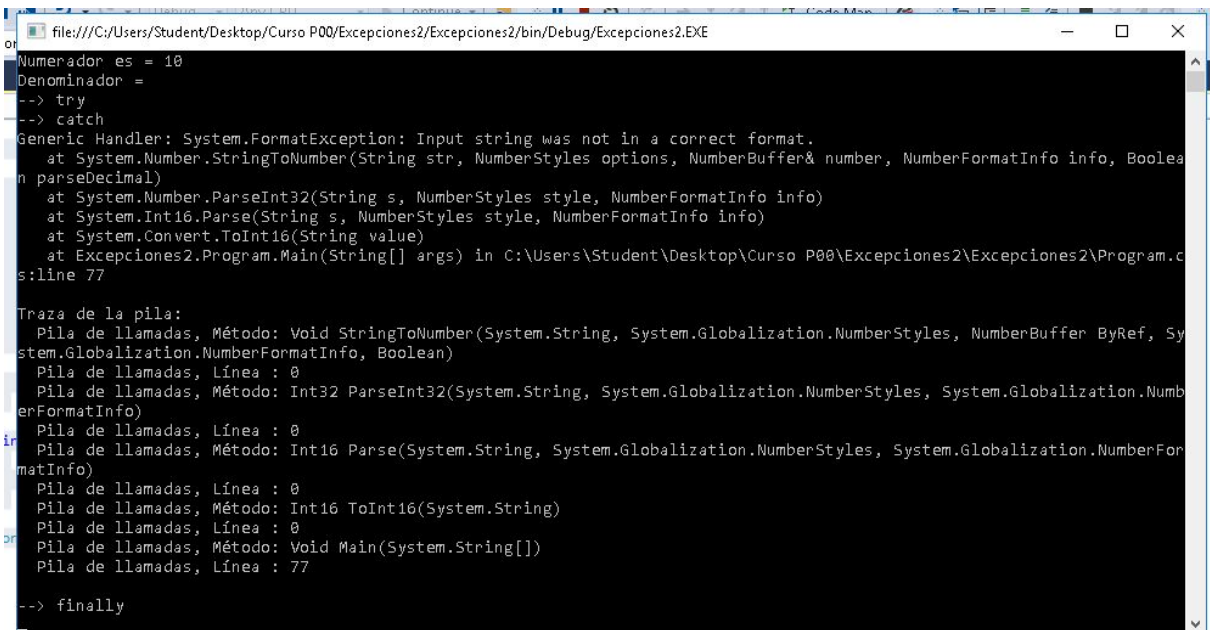
```

file:///C:/Users/Student/Desktop/Curso P00/Excepciones2/Excepciones2/bin/Debug/Excepciones2.EXE
Numerador es = 10
Denominador = 0
--> try
--> catch
Generic Handler: System.DivideByZeroException: Attempted to divide by zero.
   at Excepciones2.Program.Main(String[] args) in C:\Users\Student\Desktop\Curso P00\Excepciones2\Excepciones2\Program.cs:line 78

Traza de la pila:
   Pila de llamadas, Método: Void Main(System.String[])
   Pila de llamadas, Línea : 78
--> finally

```

3. Pruebe la ejecución **ingresando una cadena vacía como denominador**.



```

file:///C:/Users/Student/Desktop/Curso P00/Excepciones2/Excepciones2/bin/Debug/Excepciones2.EXE
Numerador es = 10
Denominador = 
--> try
--> catch
Generic Handler: System.FormatException: Input string was not in a correct format.
   at System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer& number, NumberFormatInfo info, Boolean parseDecimal)
   at System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)
   at System.Int16.Parse(String s, NumberStyles style, NumberFormatInfo info)
   at System.Convert.ToInt16(String value)
   at Excepciones2.Program.Main(String[] args) in C:\Users\Student\Desktop\Curso P00\Excepciones2\Excepciones2\Program.cs:line 77

Traza de la pila:
   Pila de llamadas, Método: Void StringToNumber(System.String, System.Globalization.NumberStyles, NumberBuffer ByRef, System.Globalization.NumberFormatInfo, Boolean)
   Pila de llamadas, Línea : 0
   Pila de llamadas, Método: Int32 ParseInt32(System.String, System.Globalization.NumberStyles, System.Globalization.NumberFormatInfo)
   Pila de llamadas, Línea : 0
   Pila de llamadas, Método: Int16 Parse(System.String, System.Globalization.NumberStyles, System.Globalization.NumberFormatInfo)
   Pila de llamadas, Línea : 0
   Pila de llamadas, Método: Int16 ToInt16(System.String)
   Pila de llamadas, Línea : 0
   Pila de llamadas, Método: Void Main(System.String[])
   Pila de llamadas, Línea : 77
--> finally

```

Ejercicio 2: Aplicar distintos tipos de excepciones

Para realizar este ejercicio, se sugiere crear una aplicación web con una página similar a la siguiente.



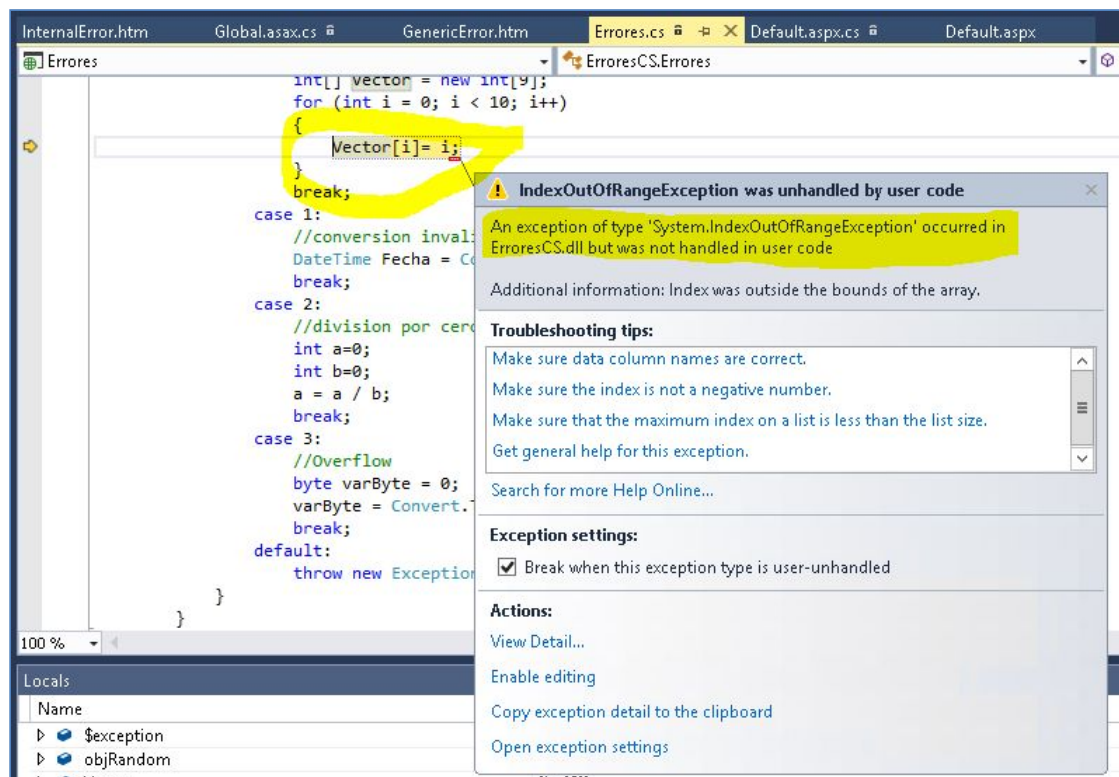
La misma tendrá tres botones:

1. El primer botón generará un error a nivel página que no será controlado con un try/catch.
2. El segundo botón generará un error a nivel página que será controlado con un try/catch.
3. El tercer botón generará un error de IIS.

Al ir realizando la ejercitación cuando programa cada caso, se recomienda ir siguiendo el código con el debugger para comprender cómo funciona el try/catch.

Mencionamos algunos detalles a tener en cuenta en el código de la aplicación. Traté de razonar la lógica del código a escribir. El proyecto con la solución completa está en la sección de descargas de esta clase.

- a. Cómo el código del primer botón no controla la excepción, .Net muestra la información completa de la excepción ocurrida. En el proyecto existe una clase con un método que genera al azar algún error en tiempo de ejecución:



- b. Cómo el código del segundo botón controla la excepción, el mismo al tener el try/catch, transfiere la ejecución a una página que muestra el mensaje de error:





```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="MostrarError.aspx.cs" Inherits="Errores.MostrarError" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h1>Se ha producido un error</h1>
            <asp:Label ID="lblMensaje" runat="server" Text=""></asp:Label>
            <br />
            <br />
            <asp:Button ID="btnVolver" runat="server" Text="Volver" OnClick="btnVolver_Click" />
        </div>
    </form>
</body>
</html>
```

- c. El tercer botón navega a una página que no existe, y al no tener try/catch, ocurre un error de IIS:



Ejercicio 3: Control de excepciones en el proyecto integrador

Basándose en el ejercicio anterior y el proyecto Errores.zip, agregue al proyecto integrador las páginas y código necesario para contemplar errores inesperados y de IIS:

- En el archivo **web.config** deberá agregar **customerrors** para redirigir la ejecución a la página **Error.html**.

```
<customErrors defaultRedirect="Error.html" mode="On"></customErrors>
```

- La página **Error.html** tendrá un código similar a este:

```
Error.html  + X
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
<link href="Content/bootstrap.css" rel="stylesheet" />
</head>
<body>
<br />
<div class="panel panel-danger">
<div class="panel-heading">Se ha producido un error</div>
<div class="panel-body">
Error inesperado.
</div>
</div>
</body>
</html>
```

- En el archivo **global.asax** en el evento **Application_Error**, deberá agregar código que tome el último error de aplicación ocurrido y redirija la ejecución a una página **error.aspx**.

```
protected void Application_Error(object sender, EventArgs e)
{
    //Captura los errores que no han sido capturados con try/catch o con el evento Page_Error
    Exception ex = Server.GetLastError();
    Server.ClearError();
    Server.Transfer("~/Error.aspx?ex=" + ex.Message);
}
```

```
Error.aspx  + X Global.asax.cs Web.config Error.html
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Error.aspx.cs" Inherits="IntegradorASP.Error">

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<link href="Content/bootstrap.css" rel="stylesheet" type="text/css" />
</head>
<body>
<form id="form1" runat="server">
<br />
<div class="panel panel-danger" >
<div class="panel-heading">Se ha producido un error</div >
<div class="panel-body">
<asp:Label ID="lblError" runat="server" Text="Error inesperado."></asp:Label>
</div>
<div class="panel-footer text-right">
<asp:Button ID="btnAceptar" runat="server" Text="Aceptar" PostBackUrl="~/Default.aspx" />
</div>
</div>
</form>
</body>
</html>
```

- Observe también el tratamiento de excepciones entre las llamadas a las distintas capas con **Try/Catch**, el bloque **Finally** y la sentencia **Throw** para relanzar una excepción a la capa superior.