BOURGEOIS Charles
StJoSup, Le Havre

SIO2

Compte-rendu
Exceptions

2022

Table des matières

1) Qu’est-ce qu’une exception ?

Une exception est un objet créé lors de situations d’erreurs, elle est traitée dans différentes sections qui leurs sont réservées.

C’est un événement qui apparaît pendant le déroulement d’un programme et qui empêche la poursuite normale de son exécution, elle peut être de toute sorte : une base de données inaccessible, un problème de type d’une variable, un fichier non accessible….

Nous utilisons l’instruction try…catch pour pouvoir « remarquer » et définir une réponse si l’une des instructions de notre programme provoque une exception.

Try permet de se terminée si succès ou bien de lever une exception.
Catch permet elle de trouver et de traiter les exceptions.

L’exception peut être traitée localement au code exécute ou transmise à la fonction appelante, qui elle-même peut choisir ce qu’elle en fait.

2) Définition des exceptions sous forme algorithmiques

Nous avons vu au cours du TP, différentes formes d'exceptions, toutes dans un cas adapté.

1 : Cas où l'on veut convertir un string en entier :

```
try
{
    string chaine = "dix";
    int valeur = Convert.ToInt32(chaine);
}
catch (Exception)
{
    Console.WriteLine("Une erreur s'est produite dans la tentative de conversion");
}
```

2 : Nous pouvons avec les exceptions connaître le message d'erreur, la pile d'appel et le type de l'exception :

```
try
{
    string chaine = "dix";
    int valeur = Convert.ToInt32(chaine);
}
catch (Exception ex)
{
    Console.WriteLine("Il y a un eu une erreur, plus d'informations cidessous :");
    Console.WriteLine();
    Console.WriteLine("Message d'erreur : " + ex.Message);
    Console.WriteLine();
    Console.WriteLine("Pile d'appel : " + ex.StackTrace);
    Console.WriteLine();
    Console.WriteLine("Type de l'exception : " + ex.GetType());
}
```

```
Il y a un eu une erreur, plus d'informations cidessous :

Message d'erreur : Input string was not in a correct format.

Pile d'appel :    at System.Number.ThrowOverflowOrFormatException(ParsingStatus status, TypeCode type)
   at System.Number.ParseInt32(ReadOnlySpan`1 value, NumberStyles styles, NumberFormatInfo info)
   at System.Convert.ToInt32(String value)
   at ExceptionTest.Program.Main(String[] args) in C:\Users\driti\OneDrive\Bureau\BTS SIO 2ème année\SLAM\M.MERCY\ExceptionTest
\ExceptionTest\Program.cs:line 24

Type de l'exception : System.FormatException
```

3 : Cas où l'on a une exception *Array*, c'est-à-dire que l'on sort d'un tableau dans le but d'obtenir une valeur à un index plus

grand que la taille du tableau :

```
try
{
    int[] tab = new int[5] { 1, 2, 3, 4, 5 };
    Console.WriteLine(tab[6]);
}
catch (Exception ex)
{
    Console.WriteLine("Message d'erreur : " + ex.Message);
    Console.WriteLine();
    Console.WriteLine("Type de l'erreur : " + ex.GetType());
}
```

```
Message d'erreur : Index was outside the bounds of the array.
```

4 : Avec une valeur null :

```
try
{
    string id = null;
    int ans = int.Parse(id);
}
catch (Exception ex)
{
    Console.WriteLine("Message d'erreur : " + ex.Message);
    Console.WriteLine();
    Console.WriteLine("Type de l'erreur : " + ex.GetType());
}
```

5 : Nous pouvons aussi remarquer une exception lorsqu'une racine carrée est négative :

```
try
{
    double racine = RacineCarree(5);
    Console.WriteLine(racine);
}
catch (Exception ex)
{
    Console.WriteLine("Impossible d'effectuer le calcul : " +ex.Message);
}
```

```
 if (valeur <= 0)
     throw new ArgumentOutOfRangeException("valeur", "Le paramètre doit être positif");
 return Math.Sqrt(valeur);
```

```
Impossible d'effectuer le calcul : Le paramètre doit être positif (Parameter 'valeur')
```

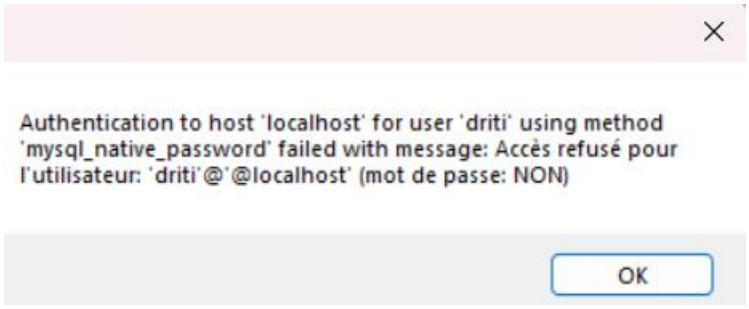
6 : Nous pouvons aussi établir un *finally*, qui même lorsqu'une exception est présente, assure l'exécution du bloc :

```
1 référence
private void Button1_Click(object sender, EventArgs e)
{
    MySqlConnection MySqlCo = new MySqlConnection();
    MySqlCo.ConnectionString = "server = localhost; userid=" + IdBOX.Text + ";password=" + IdMDP.Text + ";database=sakila";
    try
    {
            MySqlCo.Open();
            MessageBox.Show("Connexion réussie !");
            Selection OBJ = new Selection();
            OBJ.Show();
            this.Hide();

    }
    catch (Exception ex)
    {
            MessageBox.Show(ex.Message);
            MessageBox.Show("Connexion pas réussie !");
    }

    finally
    {
        MySqlCo.Close();
        this.Close();
    }
}
```

Dans notre cas lorsque notre id et mot de passe n'est pas bon, nous ne pouvons pas nous connecter dans notre base de données.



7 : Maintenant, dans un cadre plus professionnel, notre chef nous demande de lui envoyer par mail dans un fichier txt tout exception provoquée dans notre programme. Pour ce faire il nous faut une méthode EcrireDansFichierTxt puis une méthode pour lui envoyer :

```
1 référence
public static void EnvoyerMail()
{
    MailMessage msg = new MailMessage();

    msg.From = new MailAddress("charles.bourgeois@stjosup.com", "charles bourgeois");
    msg.To.Add(new MailAddress("valentin.kong@stjosup.com"));
    msg.Subject = "exception";
    msg.Body = "voici le fichier";

    msg.Attachments.Add(new Attachment("C:/Users/driti/OneDrive/Bureau/test.txt"));

    SmtpClient smtp = new SmtpClient("ssl0.ovh.net");
    smtp.Port = 587;
    smtp.Credentials = new System.Net.NetworkCredential("charles.bourgeois@stjosup.com", "BC08102003");
    smtp.EnableSsl = true;

    smtp.Send(msg);
}
```

```
1 référence
public static void EnregistrerErreurDansUnFichierDeLog(Exception ex)
{
    try
    {
        StreamWriter sw = new StreamWriter("C:/Users/driti/OneDrive/Bureau/test.txt");
        sw.WriteLine(ex);
        sw.Close();
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception: " + e.Message);
    }
    finally
    {
        Console.WriteLine("Executing finally block.");
    }

}
```

```
0 références
public static void MaMethode()
{
    try
    {
        Console.WriteLine("Veuillez saisir un entier :");
        string chaine = Console.ReadLine();
        int entier = Convert.ToInt32(chaine);
    }
    catch (Exception ex)
    {
        Console.WriteLine("La saisie n'est pas un entier");
        EnregistrerErreurDansUnFichierDeLog(ex);
        EnvoyerMail();
        throw;
    }
```

```
0 références
static void Main(string[] args)
{
    Produit produit = new Produit("guitare",50);
    try
    {
        produit.ChargerProduit("TV HD");
    }
    catch (ProduitNonEnStockException ex)
    {
        Console.WriteLine("Erreur : " + ex.Message);
    }

}
```

3) Conclusion

En programmation, la gestion des exceptions est indispensable, pour repérer et traiter des erreurs et ne pas bloquer son programme inutilement alors que l'erreur peut être visible, même envoyé par mail à un supérieur !

De plus, le bloc d'instructions try..catch…finally permet un usage aisé de ces exceptions et donc de résoudre les problèmes dans les moindres délais.