

L' héritage en Asp.net pour la gestion d'erreurs

par Ludovic Lefort ([Site web](#)) ([Blog](#))

Date de publication : 27/08/2007

Ce document prouve l'utilité de l'héritage de page dans le développement web. La gestion d'erreur peut se faire depuis le fichier Global.asax, dans ce document elle permet juste d'illustrer l'intérêt de l'héritage par un exemple simple et parlant.

- 0 - Introduction
- 1 - Création de la page parente
- 2 - Création de la page d'erreur
- 3 - Création des autres pages du site
- 4 - Conclusion

0 - Introduction

Etant de formation plutôt un développeur Windows Forms, je suis habitué à utiliser l'héritage de form dès que possible afin de limiter la redondance dans mon code.

Je suis depuis peu amené à développer des sites asp.net et à reprendre du code d'autres développeurs. Une chose m'a frappé : l'héritage au niveau des pages n'est quasiment jamais utilisé.

Asp.net est un langage orienté objet alors pourquoi se priver de la puissance de l'héritage ?

Un exemple très concret de la manière dont l'héritage peut nous aider est la gestion d'erreur. Autrement dit : comment être sûr que si une erreur non prévue survient dans le code, l'utilisateur sera redirigé vers une page "propre" et que l'erreur sera loggée et envoyée par mail à un administrateur par exemple.

1 - Création de la page parente

La page parente est celle qui contiendra le code récupérant l'erreur et qui redirigera l'utilisateur vers la page d'erreur.

Nous n'allons pas créer une page aspx mais simplement une classe .net qui héritera de **System.Web.UI.Page**.

Nous allons placer cette classe dans le répertoire appCode de notre site et l'appeler BasePage.cs

Voici le contenu de cette page :

```
public class BasePage : System.Web.UI.Page
{
    public BasePage()
    {
    }

    protected override void OnError(EventArgs e)
    {
        Exception ex = Server.GetLastError();
        Session["LastError"] = ex;
        Response.Redirect("~/Error.aspx");
    }
}
```

Notre classe hérite donc de **System.Web.UI.Page** ce qui nous permet de travailler comme à l'intérieur d'un fichier .aspx

Dans cet exemple nous allons override l'event handler **OnError** qui est déclenché lorsque une erreur non gérée est interceptée

la ligne :

```
Exception ex = Server.GetLastError();
```

permet de récupérer la dernière exception renvoyée par la système, donc celle qui a provoqué l'exécution de ce code.

Nous allons ensuite mettre cette erreur dans la variable de session afin de pouvoir la récupérer dans une autre page.

Pour finir nous allons rediriger l'utilisateur vers une page d'erreur que nous allons écrire.

2 - Création de la page d'erreur

Nous allons maintenant ajouter une page à la racine de notre site que nous allons appeler : **Error.aspx**

Cette page contient juste un label nommé **lblError**.

Voici le contenu de la page :

```
public partial class Error : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            if (Session["LastError"] != null)
            {
                Exception ex = (Exception)Session["LastError"];
                lblError.Text = ex.Message;

                //Ecriture de l'erreur dans un table LOG de la db
                //.....
                //.....

                //Envoi d'un mail aux administrateurs du site avec le contenu du ex.stackTrace()
                //.....
                //.....
            }
        }
    }
}
```

Décrivons ce code :

Tout d'abord je teste si nous ne sommes pas dans un postback pour ne pas exécuter le code plusieurs fois inutilement.

Ensuite je teste si la variable de session contient bien un élément LastError dans le cas où l'utilisateur arriverait directement sur cette page sans qu'une exception soit lancée.

L'exception que l'on a mise dans la variable de Session à l'étape précédente est récupérée et affichée dans notre label.

L'erreur est loggée dans la db et un mail est envoyé aux administrateurs du site.

3 - Création des autres pages du site

Il nous reste maintenant à créer les autres pages du site qui hériteront de **BasePage.cs**

Pour cet exemple nous allons modifier la page d'accueil de notre site : **default.aspx**

Première étape : il faut que la page hérite de **BasePage.cs**:

```
public partial class _Default : BasePage
```

Ensuite nous allons ajouter un bouton sur la page qui provoquera une exception :

```
protected void btnException_Click(object sender, EventArgs e)
{
    //Division par zéro
    int a = 10;
    int b = 0;
    int c = a / b;
}
```

La division par zéro étant mathématiquement impossible une exception sera lancée

Comme notre page hérite de BasePage.cs et que nous n'avons pas catché l'erreur l'événement `OnError()` sera déclenché et traité par la classe parente (**basepage.cs**)

4 - Conclusion

Voilà comment en quelques lignes éviter que l'utilisateur se retrouve devant une page d'erreur asp.net illisible et pas très professionnel.

Nous avons utilisé l'héritage dans cet exemple pour la gestion d'erreurs mais cela peut également permettre de coder la gestion des rôles pour tout votre site pour exemple

