Dotnet France
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

Gestion des erreurs dans les pages ASP .NET Ajax

Version 1.0

James RAVAILLE
http://blogs.dotnet-france.com/jamesr

Sommaire

1 Introduction

1.1 Présentation

Comme présenté dans le chapitre sur les bases fondamentales d'ASP .NET Ajax, il est possible, via l'utilisation du contrôle *UpdatePanel*, de mettre à jour des parties de pages. Nous vous proposons maintenant de voir, comment intercepter et gérer les erreurs, survenant pendant ces mises à jour.

1.2 Pré-requis

Avant de lire ce cours, il est nécessaire d'avoir lu le cours sur les bases fondamentales d'ASP .NET Ajax.

2 Présentation du comportement de base

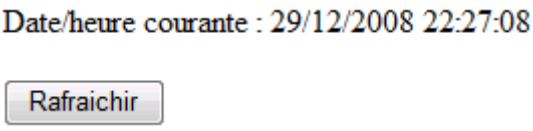
Avant de comprendre comment gérer les erreurs lors de la mise à jour partielle des pages, il convient de s'intéresser à la manière dont elles sont gérées par défaut.

Soit une page ASP .NET contenant un contrôle *ScriptManager*, un contrôle *UpdatePanel*, lui-même contenant un contrôle de type *Label* et un autre de type *Button*. L'évènement *Click* du bouton a été implémenté, de manière à afficher dans le contrôle de type *Label*, l'heure et la date courante :

```
// C#

<form id="FormAjax" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>

    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            Date/heure courante : <asp:Label ID="LblDateHeureCourante"
runat="server" Text=""></asp:Label>

            <br />
            <br />

            <asp:Button ID="CmdRafraichir" runat="server"
Text="Rafraichir"
                onclick="CmdRafraichir_Click" />
        </ContentTemplate>
    </asp:UpdatePanel>
</form>



protected void CmdRafraichir_Click(object sender, EventArgs e)
{
    LblDateHeureCourante.Text = DateTime.Now.ToString();
}
```

```
' VB.NET

<form id="FormAjax" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>

    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            Date/heure courante : <asp:Label ID="LblDateHeureCourante"
runat="server" Text=""></asp:Label>

            <br />
            <br />

            <asp:Button ID="CmdRafraichir" runat="server"
Text="Rafraichir" />
        </ContentTemplate>
    </asp:UpdatePanel>
</form>


Protected Sub CmdRafraichir_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles CmdRafraichir.Click
    LblDateHeureCourante.Text = DateTime.Now.ToString()
End Sub
```

On obtient alors le résultat suivant :

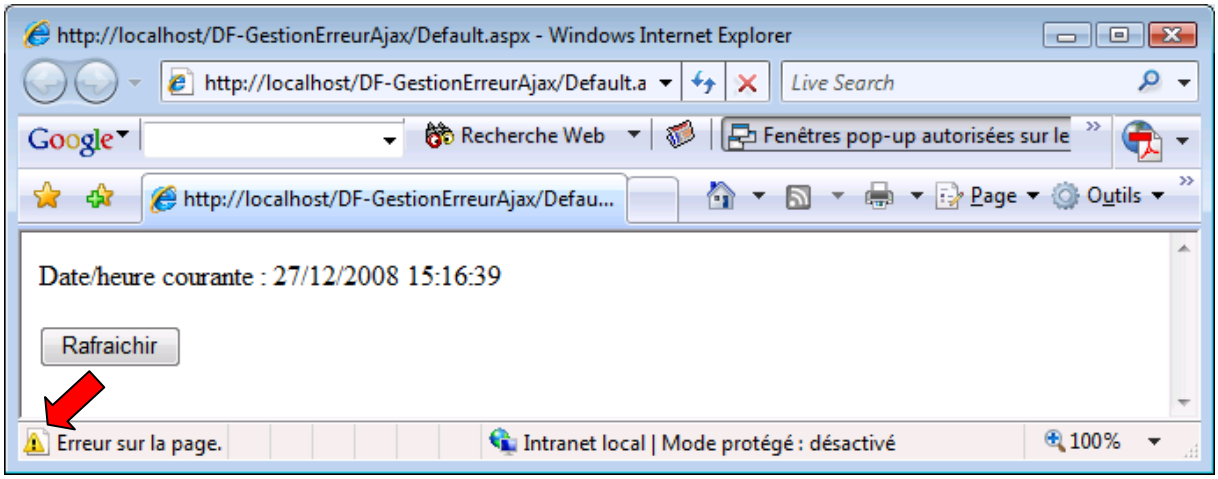


Maintenant, dans ce code Si dans ce code, au lieu d'afficher la date et l'heure courante, on lève une exception, de manière à provoquer explicitement une erreur :

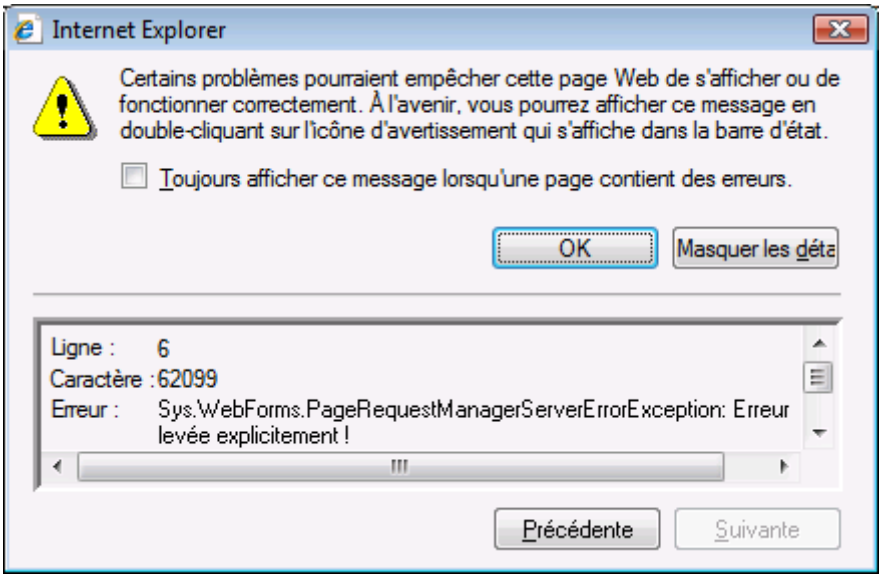
```
// C#

protected void CmdRafraichir_Click(object sender, EventArgs e)
{
    //LblDateHeureCourante.Text = DateTime.Now.ToString();
    throw new Exception("Erreur levée explicitement !");
}
```

```
' VB.NET

Protected Sub CmdRafraichir_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles CmdRafraichir.Click
    'LblDateHeureCourante.Text = DateTime.Now.ToString()
    Throw New Exception("Erreur levée explicitement !")
End Sub
```

Alors, lors de l'exécution de la page côté client, dans Internet Explorer, on obtient l'erreur suivante :



Nous conviendront que ce message d'erreur n'est pas tellement explicit. En double-cliquant sur l'erreur signalée en bas à gauche, la fenêtre suivante apparaît :



Pour gérer les erreurs, plusieurs possibilités s'offrent à nous. Nous en retiendrons deux :

- Utiliser la gestion des erreurs personnalisées des applications ASP .NET (section *customErrors* du fichier de configuration de l'application Web)
- Utiliser la gestion des erreurs proposées par ASP .NET Ajax, permettant :
  - De personnaliser la gestion et le message de l'erreur côté serveur.
  - De traiter et afficher l'erreur côté client.

3 Implémenter la gestion personnalisée ASP .NET

3.1 Implémentation de la gestion des erreurs personnalisée

Cette gestion d'erreur consiste à déclarer dans le fichier de configuration de l'application Web, une page à exécuter en fonction du code retour HTTP, de la réponse que le serveur renverra au client.

Le code retour HTTP qui nous intéresse est le code 500, autrement dit celui qui indique qu'une erreur d'exécution est survenue lors du traitement de la requête sur le serveur Web. Ainsi, dans le fichier de configuration, on utilisera la section *customErrors*, de la manière suivante :

```
<customErrors mode="On">
    <error statusCode="500" redirect="PageErreur.aspx" />
</customErrors>
```

3.2 Autorisation de la gestion des erreurs personnalisée

Cette action est nécessaire, mais pas suffisante. Il est nécessaire d'autoriser la gestion des erreurs personnalisées ASP .NET, de manière à ce qu'elle « prenne le dessus », sur la gestion des erreurs propre à ASP .NET Ajax. Pour ce faire, il faut valoriser la propriété *AllowCustomErrorsRedirect* à *true* (*false* étant la valeur par défaut) :

```
<asp:ScriptManager ID="ScriptManager1" runat="server"
AllowCustomErrorsRedirect="true">
</asp:ScriptManager>
```

4 Implémenter la gestion des erreurs en Ajax

4.1 Personnaliser le message d'erreur et traiter l'exception côté serveur

Il est possible d'intercepter l'erreur côté serveur, en implémentant l'évènement `AsyncPostBackError` sur le contrôle *ScriptManager* :

```
// C#

protected void ScriptManager1_AsyncPostBackError(object sender,
AsyncPostBackErrorEventArgs e)
{
    ScriptManager1.AsyncPostBackErrorMessage = "Une erreur est survenue
lors de la mise à jour partielle de la page : " + e.Exception.Message;
}
```

```
' VB.NET

Protected Sub ScriptManager1_AsyncPostBackError(ByVal sender As Object,
ByVal e As System.Web.UI.AsyncPostBackErrorEventArgs) Handles
ScriptManager1.AsyncPostBackError
    ScriptManager1.AsyncPostBackErrorMessage = "Une erreur est survenue
lors de la mise à jour partielle de la page : " + e.Exception.Message
End Sub
```

Cette action n'est pas obligatoire. Par défaut, la propriété *AsyncPostBackErrorMessage* du contrôle *ScriptManager*, est valorisée avec le message de l'exception levée. Cette implémentation permet alors de personnaliser le message d'erreur, ou d'effectuer toute autre action nécessaire au traitement de l'exception levée (côté serveur bien sûr).

4.2 Gérer et afficher l'erreur côté client

Dans notre page ASP .NET, ajoutons un autre un contrôle de type *Label* et nommé LblMessageUtilisateur. Ce contrôle contiendra le message d'erreur :

```
// VB .NET et C#

<form id="FormAjax" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"
        onasyncpostbackerror="ScriptManager1_AsyncPostBackError">
    </asp:ScriptManager>

    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            Date/heure courante : <asp:Label ID="LblDateHeureCourante"
runat="server" Text=""></asp:Label>

            <br />
            <br />

            <asp:Button ID="CmdRafraichir" runat="server"
Text="Rafraichir"
                onclick="CmdRafraichir_Click" />
        </ContentTemplate>
    </asp:UpdatePanel>

    <br />
    <br />

    <asp:Label ID="LblMessageUtilisateur" runat="server"
Text=""></asp:Label>
</form>
```

Pour gérer les erreurs côté client (dans notre cas, afficher le message dans le contrôle *Label* ajouté), il est nécessaire d'ajouter un bloc de code JavaScript dans la page, qui réaliserait les tâches suivantes :

- Lors du chargement de la page côté client, demander à exécuter une fonction JavaScript, après l'exécution de toute requête HTTP envoyée par le client.
- Cette fonction JavaScript, doit vérifier si une erreur existe. Si tel est le cas, afficher ce message, dans le contrôle *LblMessageUtilisateur*.

```
// VB .NET et C#

<script type="text/javascript" language="javascript">
    function pageLoad()
    {
        var manager = Sys.WebForms.PageRequestManager.getInstance();
        manager.add_endRequest(endRequest);
    }

    function endRequest(sender, args)
    {
        if (args.get_error() != null) {
            $get("LblMessageUtilisateur").innerHTML =
args.get_error().message;
            args.set_errorHandled(true);
        }
    }
</script>
```

L'instruction *args.set_errorHandled(true);* permet de signifier au navigateur, que l'erreur rencontrée a été traitée.

Ainsi, on obtient le résultat suivant :