# ASP.NET Fondamental

Evènements de page

WAVENET
Intelligent IT Solutions

# Evènements de page

- **Preinit**
  - Déclenché à l'issue de l'étape de démarrage et avant le début de l'étape d'initialisation.
  - Utilisez cet événement dans les cas suivants :
    - Vérifier la propriété IsPostBack pour déterminer si elle est traitée pour la première fois. Les propriétés IsCallback et IsCrossPagePostBack ont également été définies à ce stade.
    - Créer ou recréer des contrôles dynamiques.
    - Définir dynamiquement une page maître.
    - Définir la propriété Theme dynamiquement.
    - Lire ou définir des valeurs de propriétés de profil.
  - **Remarque**
    - Si la demande est un PostBack, les valeurs des contrôles n'ont pas encore été restaurées à partir de l'état d'affichage. Si vous définissez une propriété du contrôle à ce stade, sa valeur peut être substituée dans l'événement suivant.

# Evènements de page

- **Init**

  - Déclenché après que tous les contrôles ont été initialisés et tous les paramètres d'apparence ont été appliqués. L'événement Init de contrôles individuels se produit avant l'événement Init de la page.

  - Utilisez cet événement pour lire ou initialiser des propriétés de contrôle.

# Evènements de page

- **InitComplete**

  - Déclenché à l'issue de l'étape d'initialisation de la page. Une seule opération a lieu entre les événements Init et InitComplete :
    - le suivi des modifications de l'état d'affichage est activé. Le suivi de l'état d'affichage permet aux contrôles de rendre persistante toute valeur ajoutée par programmation à la collection ViewState. Jusqu'à ce que le suivi de l'état d'affichage soit activé, toutes les valeurs ajoutées à l'état d'affichage sont perdues lors des publications (postback). Les contrôles activent généralement le suivi de l'état d'affichage immédiatement après le déclenchement de leur événement Init.

  - Utilisez cet événement pour apporter des modifications que vous souhaitez rendre persistantes à l'état d'affichage après la prochaine publication (postback).

# Evènements de page

- **PreLoad**

  - L'événement PreLoad est déclenché après le traitement de toutes les données de publication (postback) et avant l'événement Load. Il y a une deuxième tentative de chargement des données de publication (postback) avant l'événement OnLoadComplete

  - L'événement Load de contrôles individuels se produit après l'événement Load de la page.

# Evènements de page

- **Load**
  - L'objet Page appelle la méthode OnLoad sur l'objet Page.
  - Utilisez la méthode d'événement OnLoad pour définir des propriétés dans les contrôles et établir des connexions de base de données.

# Evènements de page

- **LoadComplete**
  - Déclenché à la fin de l'étape de gestion des événements.
  - Utilisez cet événement pour les tâches qui requièrent que tous les autres contrôles de la page soient chargés.

# Evènements de page

- **PreRender**

  - Déclenché après que l'objet Page a créé tous les contrôles requis pour restituer la page, notamment les contrôles enfants des contrôles composites. (Pour ce faire, l'objet Page appelle EnsureChildControls pour chaque contrôle et pour la page.)

  - L'objet Page déclenche l'événement PreRender sur l'objet Page, puis fait de même pour chaque contrôle enfant. L'événement PreRender de contrôles individuels se produit après l'événement PreRender de la page.

  - Utilisez cet événement pour apporter des modifications finales au contenu de la page ou à ses contrôles avant le début de l'étape de rendu.

# Evènements de page

- **PreRenderComplete**

  - Déclenché après que chaque contrôle lié aux données, dont la propriété DataSourceID est définie, et ait appelé sa méthode DataBind.

# Evènements de page

- **SaveStateComplete**
  - Déclenché après que l'état d'affichage et l'état du contrôle ont été enregistrés pour la page et pour tous les contrôles. À ce stade, toute modification de la page ou des contrôles affecte le rendu, mais aucune n'est récupérée dans la publication (postback) suivante.

# Evènements de page

- **Render**

  - Il ne s'agit pas d'un événement ; au lieu de cela, l'objet Page appelle cette méthode sur chaque contrôle à ce stade du traitement. Tous les contrôles serveur Web ASP.NET ont une méthode Render qui écrit le balisage du contrôle à envoyer au navigateur.

  - Un contrôle utilisateur (un fichier .ascx) incorporant automatiquement le rendu, vous n'avez pas besoin de restituer explicitement le contrôle dans le code. ??????

# Evènements de page

- **Unload**
  - Déclenché pour chaque contrôle puis pour la page.
  - Dans les contrôles, utilisez cet événement pour effectuer un dernier nettoyage pour des contrôles spécifiques.  Par exemple, la fermeture des connexions de base de données spécifiques aux contrôles.
  - Pour la page elle-même, utilisez cet événement pour effectuer un dernier nettoyage. Par exemple, la fermeture des connexions de base de données et des fichiers ouverts, ou la fin de journalisation ou d'autres tâches spécifiques à la demande.

# ASP.NET Fondamental

Plus de contrôles ASP.NET

WAVENET
Intelligent IT Solutions

# Asp.net More Controls

- Panel
- HiddenField
- MultiView
- SiteMap
- Repeater
- GridView
- ListView

# Panel

- Il Permet d'englober un groupe de composants html ou asp
- Exemple :

```
<asp:Panel ID="MyPanel" runat="server">
    <span>
        Something inside the panel
    </span>
</asp:Panel>
```

# HiddenField

- Il génère un input de type « hidden » ce qui vous permet d'intégrer ce que vous voulez.

- Exemple :

```
<asp:HiddenField ID="HiddenField1" runat="server" Value="MyValue" />
```

# MultiView

- Il est très pratique dans le cas où vous avez un processus quelconque basé sur plusieurs vues successives comme un processus de validation d'achat par exemple.

- Exemple :

```
<asp:MultiView ID="MultiView1" runat="server">
    <asp:View ID="View1" runat="server">

    </asp:View>
    <asp:View ID="View2" runat="server">

    </asp:View>
</asp:MultiView>
```

# SiteMap

- Le plan du site permet d'afficher à l'utilisateur le chemin parcouru pour arriver à la page courante.
- Exemple :

**Home > Contact > …**

  - Contrôle asp « SiteMapPath » :

```
<asp:SiteMapPath ID="ContactSiteMapPath" runat="server"></asp:SiteMapPath>
```

  - Fichier « web.sitemap » :

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="/" title="Home"  description="Home">
      <siteMapNode url="/Contact.aspx" title="Contact"  description="Contact">
      </siteMapNode>
    </siteMapNode>
</siteMap>
```

WAVENET
Intelligent IT Solutions

# Repeater

- Ce contrôle permet, sur base d'un template, d'afficher une série d'éléments.

```
<asp:Repeater ID="ModelRepeater"
              runat="server">
    <ItemTemplate>
        <div>
            <%# Eval("Name") %>
        </div>
    </ItemTemplate>
</asp:Repeater>
```

```
public class Model
{
    public Model Parent { get; set; }
    public string Name { get; set; }
}

public partial class Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            Model m1 = new Model() { Name = "Model 1" };
            Model m2 = new Model() { Name = "Model 2" };
            Model m3 = new Model() { Name = "Model 3" };
            Model m4 = new Model() { Name = "Model 4" };
            Model m5 = new Model() { Name = "Model 5" };

            ModelRepeater.DataSource = new[] { m1, m2, m3, m4, m5 };
            ModelRepeater.DataBind();
        }
    }
}
```

# Exercice

- Créer un site de vente de produits simples :
- Chaque produit à :
  - Un Id
  - Un Nom
  - Une Description
  - Un Prix
- Respecter cette structure de page :