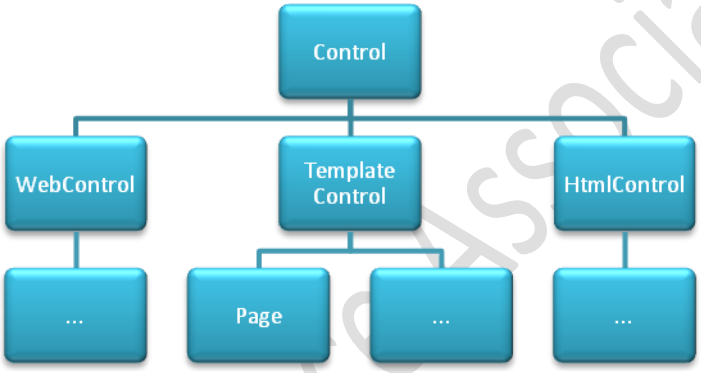
Contrôle serveur

1 Principes du contrôle serveur

1.1 Définition

Un contrôle serveur est une balise (HTML ou ASP) associé à la propriété `runat="server"`. A cette balise sera associé un objet lors de l'exécution du code par le serveur, il sera donc instancié au chargement de la page et peut être appelé par le code behind grâce à son `id`. Un contrôle serveur est programmable depuis le code behind pour répondre à des évènements (comme un clic, le chargement du contrôle et d'autres).

Les Contrôles sont des classes héritant de la classe Control comme le présente ce diagramme (non exhaustif et vous n'êtes pas obligé de connaitre les diagrammes d'héritage, ils sont là juste à titre informatif) :



Les WebControl et HtmlControl seront développés ultérieurement.

La classe Control contient un attribut appelé « ViewState ». C'est un dictionnaire d'état des objets, ainsi tous les états des contrôles sont sauvegardés (dans un champ caché qui est visible dans la source HTML en chiffré). L'intérêt ? ASP.NET, à partir de ça, restaure automatiquement les états des contrôles seulement quand la page est rechargée. On appelle ce phénomène le PostBack.

Le ControlState est similaire au ViewState, il permet de le désactiver globalement (le ViewState est assez gourmand en ressources) sans perdre les informations de rendu ou autres informations importantes à faire passer pour l'affichage minimal.

Pour désactiver le ViewState in faut mettre la propriété `EnableViewState` à false dans la directive de la page ASPX (ce qui donnerait sans les autres propriétés de la directive : `<%@ Page EnableViewState="false" %>`) .

1.2 Cycle de vie de la page

La page est un cas particulier de contrôle serveur, elle a la particularité de ne pas avoir besoin de balise pour être instancié. Nous allons voir les évènements de contrôles serveur du cycle de vie de la page.

Un évènement est un moment précis dans l'exécution du code, il y a plusieurs évènements durant la vie de la page. La page effectue un cycle durant lequel elle initialise ses contrôles enfants qui à leur tour feront de même (c'est une fonction récursive qui va initialiser tous les contrôles

existant). On parle de contrôle enfant pour un contrôle qui est contenu dans un autre contrôle (qui lui sera le contrôle parent).

Durant le cycle de vie de la page, suivant le moment où l'on fait les actions il se peut que les objets ne soient pas encore instanciés ou qu'ils ne soient plus modifiables. Il est essentiel de prendre conscience de la notion d'évènement et de l'importance du choix de l'instant d'exécution des fonctions.

Pour vous aider à comprendre l'évolution d'une page ASP.NET, nous allons lister les évènements de son exécution. Ce qui suit est une liste des évènements associés à leur sens ou utilisation courante, elle n'est pas exhaustive et vous n'êtes pas tenu de la connaitre par cœur.

Phase d'initialisation :

PreInit :

- Avant tout calcul de la page (rien n'est initialisé).
- Permet de changer le thème (seul moment où l'on peut changer le css).
- Déterminer si la page est chargée pour la première fois (propriété IsPostBack du contrôle Page).

Init :

- Apparence chargée.
- Contrôles initialisés.
- Permet de lire / initialiser des propriétés de contrôle.

InitComplete

- Fin de la phase d'initialisation (tous les objets sont initialisés)

Phase de chargement :

Preload :

- A utiliser si on a besoin d'effectuer une action relative au chargement avant l'évènement Load.

Load :

- Charge récursivement les contrôles enfants de la page

Load Complete :

- Pour les actions nécessitant que l'ensemble de la page soit chargée

Phase de rendu :

Prerender :

- Dernier évènement qui permet de modifier le contenu de la page et des contrôles
- Permet de remplir les contrôles avec des données s'il y a lieu (par rapport à une source de données telle qu'une base SQL, un fichier XML …)

SaveStateControl :

- A partir de cet évènement, toute modification de la page et/ou de ses contrôles sera ignorée.
- Permet de sauvegarder l'état des contrôles tels qu'ils seront affichés.

Render :

- Génération du code HTML pour le navigateur.
- Render n'est pas un évènement (c'est une méthode).

Phase de fermeture :

Unload :

- Utilisé pour fermer les connexions à toutes les sources de données et/ou vider les variables temporaires en fin vie de la page.

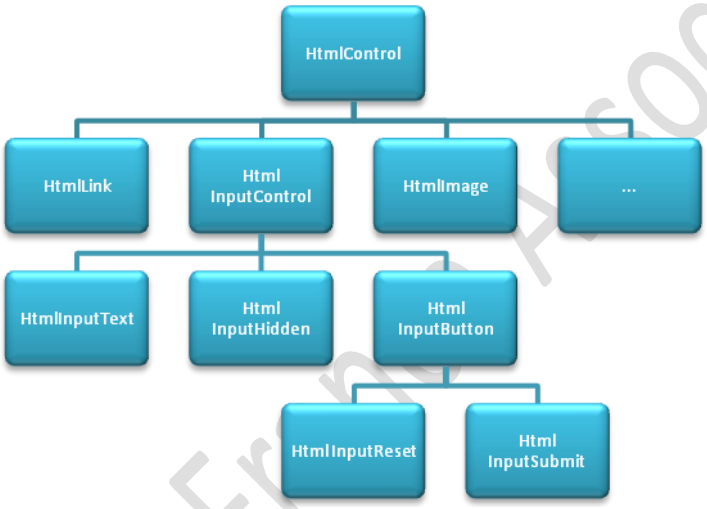
Remarque :

Les méthodes associées à chaque événement sont le nom de l'évènement précédé d'un « On » (exemple : OnInit, OnPreRender, …). Toutefois il n'existe pas de méthode pour Render car ce n'est pas un évènement mais une méthode.

2 Types de contrôles

2.1 HTML

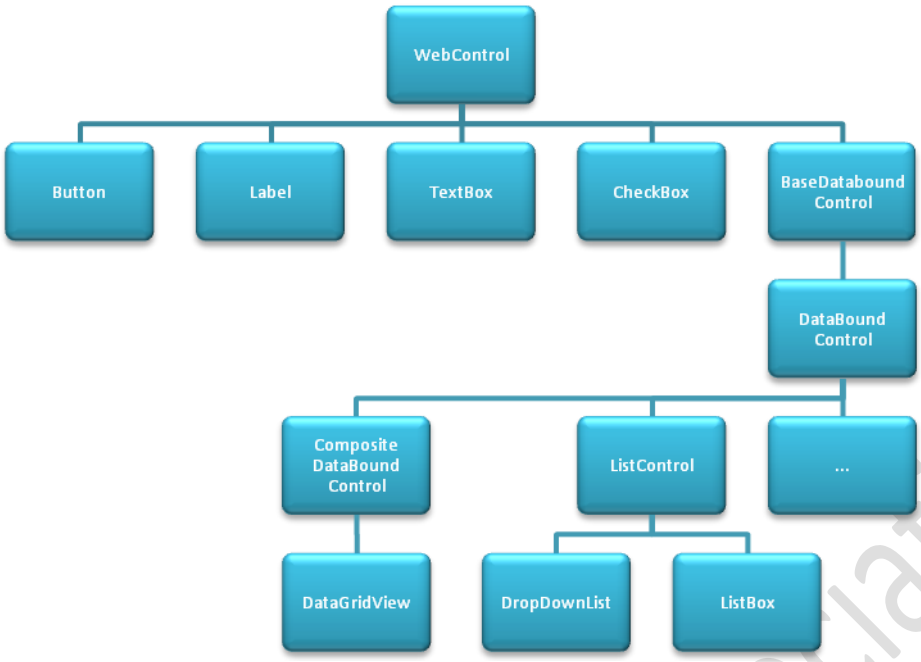
Les contrôles HTML sont des contrôles serveurs dans le cas où ils ont un attribut `runat="server"`. Vous aurez des exemples très bientôt de ces contrôles mais avant de voir le code reprenons l'arbre d'héritage des contrôles (celui de la première page) et montrons comment les contrôles ont étés fait par ressemblance.



Ces classes existent principalement pour la compatibilité avec la syntaxe HTML pour permettre de porter facilement un site statique existant en ASP.NET.

2.2 Web

Les contrôles Web sont des outils plus complets en fonctionnalités que les contrôles HTML, ils héritent d'une classe plus riche que le contrôle HTML, nous verrons les différences fonctionnelles dans la prochaine sous-partie qui montre les principaux attributs.



La syntaxe des contrôles Web est :

```
<asp:NomControle id="nom_de_mon_controle" runat="server"></asp:NomControle>.
```

2.3 Principaux attributs

2.3.1 Communs

Nous allons vous présenter les attributs les plus utiles qui existent autant pour les contrôles serveur Web que HTML.

Attributs	Type	Description
Visible	Booléen	Permet d'afficher ou dissimuler un contrôle.
ID	String	Identifiant unique sert de nom d'objet pour l'appel dans le code behind.
Parent	Control	Pointe sur le contrôle parent.

2.3.2 Distincts

Voici les attributs qui diffèrent suivant que ce soit un contrôle serveur Web ou HTML.

2.3.2.1 Web

Attributs	Type	Description
CSS	-	Pour des raisons pratiques les contrôles Web ont des attributs pour accéder facilement aux propriétés CSS dont : BackColor, BorderColor, BorderStyle, Font, Width, Height
CssClass	String	Correspond à l'attribut `class` en HTML
AccessKey	String	Pour associer un raccourci clavier à un contrôle ex
TabIndex	Integer	Permet à l'utilisateur de sélectionner dans un ordre prédéfini par les valeurs (du plus petit au plus grand) en se déplaçant avec la touche TAB.

Enabled	Booléen	Permet d'interdire les interactions avec un contrôle (grise les champs). (Inutile sur un label)
ToolTip	String	Permet d'afficher une info-bulle.

2.3.2.2 HTML

Attributs	Type	Description
Name	String	Donne une appellation utile pour le code non générés par visual studio comme du JavaScript et la rétrocompatibilité.
Disabled	Booléen	Correspond à l'attribut `class` en HTML
Value	String	Valeur retournée par défaut par le contrôle HTML.

2.4 Comment changer les attributs d'un contrôle serveur

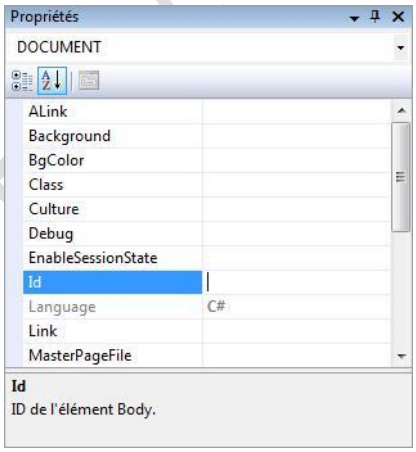
Il existe trois façons de modifier les attributs / propriétés d'un contrôle serveur que nous allons vous présenter dans les parties qui suivent.

2.4.1 A partir du code de la page ASPX

Il suffit de rajouter sur la balise à laquelle on veut le changer l'attribut avec sa valeur. Ce sera de la forme « MonAttribut="MaValeur" ».

Exemple : `<body id="test">`

2.4.2 A l'aide de la fenêtre propriétés



Fenêtre de propriétés pour changer les attributs

Pour avoir cette fenêtre il faut avoir sélectionné un contrôle serveur. Dans cette fenêtre on peut changer tous les attributs du contrôle serveur sélectionné.

2.4.3 A partir du code behind

Avant cela il faut mettre un attribut ID sur le contrôle serveur. Un ID est unique (il ne peut y en avoir qu'un du même nom). Ainsi on pourra y accéder à partir du code behind.

Dans le code behind il faut ajouter une ligne de cette forme : MonId.MonAttribut = "MaValeur"

[Code à mettre dans la page ASPX]

```
<form runat="server">
    <asp:Button ID="Button1" runat="server" Text="Button" />
</form>
```

C#

```
Button1.ID = "test";
```

VB.NET

```
Button1.ID = "test"
```

Ici Button1 correspond à l'id du contrôle serveur, le « . » permet de dire que sa s'applique à lui (descendre dans une forme d'arborescence), le ID ="test" défini l'attribut ID avec la valeur test.

Remarque :

Ici pour accéder aux attributs de notre bouton il faudra toujours utiliser Button1 même si l'ID a été changé en cours de route.

3 Contrôles communs

Avant de parler des contrôles basiques, nous allons voir comment ajouter les contrôles serveur.

3.1 Ajouter des contrôles serveur

3.1.1 Les 3 modes

Pour ajouter un contrôle serveur (qu'il soit Web ou HTML) il est possible de faire un drag & drop depuis la toolbox. Il existe 3 modes d'affichage : fractionner, design et source et on peut faire le drag & drop dans chacun d'eux (un exemple est proposé dans le Webcast).

Les 3 différents modes d'affichage

Le mode Fractionner n'existe que depuis Visual Studio 2008.

3.1.2 Le code behind

Il est aussi possible de créer un contrôle serveur depuis le code behind. Pour l'expliquer voici un bout de code :

C#

```
TextBox MonTextBox = new TextBox();
MonTextBox.ID = "login";
MonTextBox.Visible = true;
form1.Controls.Add(MonTextBox);
```

VB.NET

```
Dim MonTextBox As New TextBox
MonTextBox.ID = "login"
MonTextBox.Visible = True
form1.Controls.Add(MonTextBox)
```

Explication : Tout d'abord on instancie un objet nommé « MonTextBox » de type TextBox. Ensuite on lui applique un ID « login » et on le rend visible.

3.2 Label :

Un contrôle Label est un contrôle qui permet d'afficher du texte sur une page web qui peut être modifié dynamiquement par le serveur. On accède au texte du label dans le code behind grâce à son `ID` et l'attribut `Text`.

Label

Image d'un Label

HTML

```
<label runat="server">Label</label>
```

Web

```
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

Remarque :

1/ Par défaut le contrôle HTML a le même comportement que la balise HTML, c'est-à-dire qu'il va permettre de lier le texte à un autre contrôle (RadioButton, un CheckBox, un TextBox …). Pour que le contrôle Web fasse la même chose, il faut lui rajouter l'attribut AssociatedControlID="CheckBox2" où CheckBox2 est l'id du contrôle à lier.

Exemple :

Avec contrôle HTML :

```
<label runat="server" for="CheckBox1">Appuyer ici</label>
      <asp:CheckBox ID="CheckBox1" runat="server" />
```

Avec contrôle Web :

```
<asp:Label ID="Label1" runat="server" Text="Appuyer ici"
AssociatedControlID="CheckBox2" />
      <asp:CheckBox ID="CheckBox2" runat="server" />
```

2/ Le contrôle serveur Web ajoute une balise dans le code HTML (le contrôle serveur HTML lui reste une balise <label></label>) et il est impossible d'appliquer un événement onclick sur un label.

3.3 Button :

Un contrôle Button est un contrôle sur lequel on peut cliquer (bouton poussoir) et qui déclenchera alors le PostBack de la page (par défaut). Après le PostBack il exécutera la méthode qui lui est associé. On peut utiliser un bouton pour valider un formulaire, mais aussi pour effectuer une tache assigné a ce bouton (redirection, ajouter un article au panier, réinitialiser un formulaire …).

Image d'un bouton

HTML

```
<input id="Button" type="button" value="button" />
```

Web

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

Remarque :

Un Button ne peut pas se trouver en dehors d'un form.

3.4 TextBox :

Un contrôle TextBox est un contrôle qui permet à l'utilisateur d'entrer du texte (c'est une zone de saisie).

Cette zone de saisie peut être SingleLine, sur une seule ligne, ou MultiLine, sur plusieurs lignes. Le contrôle Web, par défaut, est SingleLine mais admet un attribut qui le rend MultiLine. Le contrôle HTML est quand à lui spécialisé : c'est-à-dire qu'il existe un contrôle Singleline et un contrôle MultiLine. Plus précisément, le SingleLine correspond à un TextBox et le MultiLine à un Textarea. On accède a son contenu de la même manière qu'avec le Label : c'est à dire avec son id et l'attribut Text.

Image de TextBox

Image de Textarea

HTML

```
<input id="Text1" type="text" /> <%-- SingleLine --%>

<textarea runat="server">Texte</textarea> <%-- Multiline --%>
```

Web

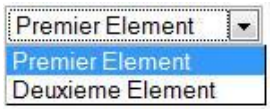
```
<asp:TextBox ID="TextBox1" runat="server" TextMode="SingleLine">
          </asp:TextBox> <%-- SingleLine --%>

<asp:TextBox ID="TextBox1" runat="server" TextMode="MultiLine">
          </asp:TextBox> <%-- MultiLine --%>
```

Remarque :

Un TextBox ne peut pas se trouver en dehors d'un form.

3.5 DropDownList :

Une DropDownList est une liste déroulante dans laquelle on ne peut sélectionner qu'un seul élément. A utiliser lorsque l'on a une grosse liste d'élément dans laquelle on ne doit pouvoir choisir qu'un élément.



DropDownList

HTML

```
<select id="Select1">
            <option>Choix 1</option>
            <option>Choix 2</option>
</select>
```

Web

```
<asp:DropDownList ID="DropDownList1" runat="server">
            <asp:ListItem Text="Choix 1" />
            <asp:ListItem Text="Choix 2" />
</asp:DropDownList>
```

Remarque :

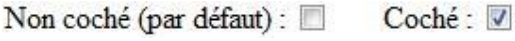
Un Button ne peut pas se trouver en dehors d'un form.

On remarque que pour l'HTML serveur il faut mettre des balises « option » entre les balises « select » pour rajouter des éléments dans la liste alors qu'en Web il faut rajouter des « ListItem » entre des « DropDownList ».

3.6 CheckBox :

Les contrôles CheckBox sont des cases à cocher qui peuvent être assimilées à un booléen dans le sens où il permet de choisir entre deux état : sélectionné ou non (true ou false). L'attribut Checked permet de définir si la case doit être cochée ou non au chargement du contrôle, donc par défaut la case n'est pas cochée.

Il est utile de remarquer que l'attribut Checked du contrôle Web prend pour valeur true ou false alors que pour le contrôle HTML il ne prend que la valeur « checked ».



Checkbox

HTML

```
<input id="Checkbox2" type="checkbox" />
```

Web

```
<asp:CheckBox ID="CheckBox1" runat="server"/> <%-- CheckBox simple --%>
```

Remarque :

Par défaut ni le contrôle Web ni le contrôle HTML n'a de texte associé. Pour cela en HTML on associe un label et un texte qui va permettre de cocher / décocher la CheckBox en cliquant juste sur le texte. Comme dans l'exemple ci-dessous.

```
<label>
      <input id="Checkbox2" type="checkbox" /> Texte
</label>
```

Ceci est une autre syntaxe de l'exemple présenté dans la partie Label.

En ce qui concerne le contrôle Web, il suffira d'ajouter un attribut Text= « ».

```
<asp:CheckBox ID="CheckBox1" runat="server" Text="Texte"
      TextAlign="Right" />
```

L'attribut TextAlign permet de définir si le texte se positionnera à gauche (left) ou à droite (right) du CheckBox.

Voici le résultat de ces deux codes :

☐ Texte

Autre remarque :

Un Button ne peut pas se trouver en dehors d'un form.

3.7 RadioButton :

Les RadioButton sont un contrôle serveur garantissant l'unicité de la sélection dans une liste de propositions. Faire attention lors de son utilisation. En effet une fois que l'on en a coché un il est impossible de décocher (il faut donc prévoir un RadioButton correspondant au cas nul/ par défaut). Aussi n'est-il pas recommandé de l'utiliser avec une liste un peu longue (car encombrant), utiliser plutôt une DropDownList dans ce cas.

RadioButton

HTML

```
<input id="Radio1" type="radio" value="maValeur" />
```

Web

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server">
      <asp:ListItem Value="maValeur" Text="texte_a_afficher" Selected="True" />
      <asp:ListItem Value="maValeur2" Text="texte_a_afficher2" Selected="False" />
</asp:RadioButtonList>
```

`Les RadioButtonList` permettent de grouper les RadioButton qui les composent. `ListItem` correspond à un RadioButton. `Value` définit la valeur de la RadioButton (par exemple « oui » ou « non »). `Text` sera le texte associé au RadioButton. `Selected` défini si le RadioBouton est coché ou non par défaut (en HTML il faut mettre `checked="checked"` pour dire qu'il est coché).

Remarque :

Par défaut le contrôle HTML ne possède pas de texte associé. Pour cela nous utiliserons un label pareillement que pour le Checkbox.

Un Button ne peut pas se trouver en dehors d'un form.

4 Evénements

4.1 Introduction

Un événement est une action (par exemple un clic sur un bouton, le changement d'un TextBox ect …) qui va pointer vers une méthode dans le code behind. Cela permet de programmer ce qui se passe lors d'un événement. Lorsque l'on défini un événement sur un contrôle on parle d'abonner le contrôle à un événement.

Exemple avec l'abonnement d'un bouton :

[Code à placer dans la page ASPX]

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Cliquer ici"
onclick="Methode_Click" />
<asp:Label ID="Label1" runat="server" Text="Vide"></asp:Label>
```

Code à placer dans le code behind :

C#

```
protected void Methode_Click(object sender, EventArgs e)
      {
            Label1.Text = TextBox1.Text;
      }
```

VB.NET

```
Protected Sub Methode_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
        Label1.Text = TextBox1.Text
End Sub
```

Explication : Le TextBox est associé à l'événement onclick. Si on clic dessus, l'événement appelle la méthode qui y est associé (en l'occurance Methode_*Click*) et éxecute le code qu'il contient. Dans cet exemple le code en question va prendre le texte qui se trouve dans le TextBox et le mettre dans le Label.

Remarque : Par défaut le nom de la méthode associé à un événement (quand l'IDE génère la méthode) est IdControle_Evenement.

4.2 Comment s'abonner ?

Il existe 2 méthodes pour abonner un contrôle à un événement : la méthode manuelle et la méthode graphique. On utilisera pour les montrer le même exemple à chaque fois (le même qu'en 4.1).

Méthode manuelle :

Dans la balise du contrôle (dans la page ASPX) on rajoute la propriété correspondant à l'événement souhaité (par exemple onclick, ontextchanged ect….) à laquelle on attribut le nom de la méthode se trouvant dans le code behind qui correspond à ce qui doit être exécuté dès que cet événement arrive.

Exemple : `<asp:Button ID="Button1" runat="server" Text="Cliquer ici" onclick="Methode_Click" />` (Ici c'est `onclick="Fonction_Click"` la partie qui a été rajouté. On c'est donc abonné à l'événement click qui pour se contrôle ira exécuter la méthode `Methode_Click` dans le code behind)

Il ne reste plus qu'a aller dans le code behind et a créer la méthode associé.

C#

```
protected void Methode_Click(object sender, EventArgs e)
    {
          Label1.Text = TextBox1.Text;
    }
```

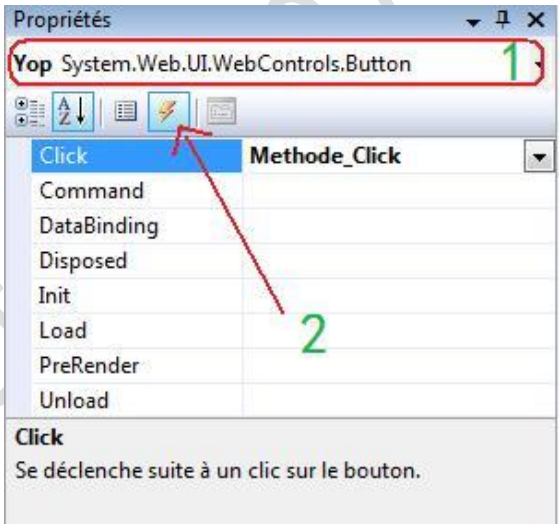
VB.NET

```
Protected Sub Methode_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
        Label1.Text = TextBox1.Text
End Sub
```

Méthode graphique :

On commence comme pour la manuelle : il faut ajouter le contrôle sur lequel on veut mettre l'événement. Pour nous ce sera le même que tout à l'heure mais sans la propriété de l'événement :

```
<asp:Button ID="Button1" runat="server" Text="Cliquer ici" />
```



Abonner un contrôle serveur à un evènement en C# (Méthode Graphique)

En C# il faut selectionner le contrôle (un clic dessus suffit) et aller dans la fenêtre « Propriétés ».

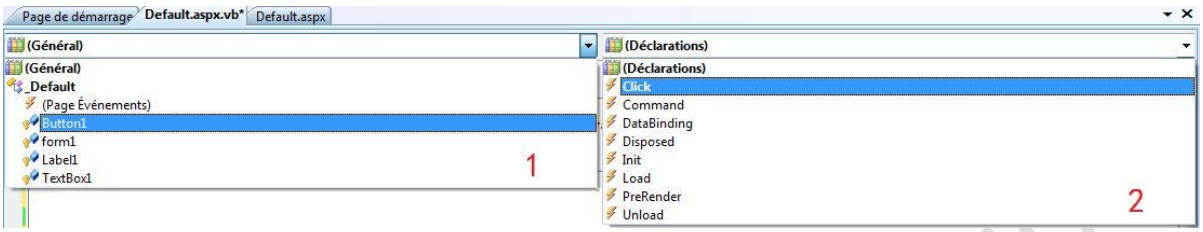
1/ Vérifier qu'est selectionné le bon contrôle (ici Yop est l'id du contrôle et la suite dis de quel type de contrôle il s'agit).

2/ Le bouton pour afficher les événements disponible sur ce type de contrôle (ici sur un button).

La liste contient tous les événements disponible … Celui qui nous interesse dans cet exemple est le

premier (Click) pour lequel on va faire passer en paramètre le nom de la méthode qui correspondra à cet événement (Page_Load).

En VB.NET on doit aller dans le code behind et en haut de la fenêtre on a ceci :



Abonner un contrôle serveur à un evènement en VB.NET (Méthode Graphique)

1/ Choix du contrôle serveur sur lequel appliquer l'événement (nous on prend TextBox1).

2/ Choix de l'événement (Click pour nous).

Une fois ceci fait Visual Studio va générer la méthode dans le code behind (que l'on soit en C# ou pas). Il ne reste donc plus qu'a le remplir.

Astuces : Vous pouvez abonner un contrôle serveur à partir du mode design en doublecliquant sur sa représentation. Seulement vous ne pouvez abonner que l'événement par défaut de ce contrôle. Par exemple l'événement click pour un bouton.