

ASP.NET V2

Sommaire des ateliers

INTRODUCTION

ATELIER 1

1 CRÉER UN PREMIER FORMULAIRE DYNAMIQUE.....3

ATELIER 2 : COMPRENDRE LA PROBLEMATIQUE DE GESTION D'ETAT

~~ATELIER 3 : UTILISER LE MODELE EVENEMENTIEL~~

PRINCIPE

TRAITER DES ÉVÈNEMENTS

AUTRE EXEMPLE D'ÉVÈNEMENTS

ATELIER 4 : COMPRENDRE L'APPROCHE COMPOSANT

INTÉRÊT DE L'APPROCHE

LE CONTRÔLE CALENDAR

COMBINER PLUSIEURS CONTRÔLES

ATELIER 5 : UTILISER VISUAL WEB DEVELOPER

... EN TANT QUE GESTIONNAIRE DE PROJET

... EN TANT QU'ÉDITEUR

... EN TANT QUE COMPILATEUR

... EN TANT QUE DÉBOGUEUR

ATELIER 6 : ACCÉDER À DES DONNÉES

SE CONNECTER À UNE SOURCE DE DONNÉES

AFFICHER DES DONNÉES

POUR ALLER PLUS LOIN

ATELIER 7 : IMPLÉMENTER LE MODÈLE DE CODE-BEHIND

ISOLER LE CODE DE LA PRÉSENTATION DANS UNE PAGE WEB

COMPILER ET DÉPLOYER LE SITE WEB

COMPRENDRE LE CYCLE DE VIE D'UNE PAGE WEB

1 Créer un premier formulaire dynamique

Dans cet exercice, vous allez apprendre à :

- Rendre dynamique un formulaire html.
- Utiliser le serveur de test de Visual Studio.
- Programmer un aller retour (post back) sur le serveur.

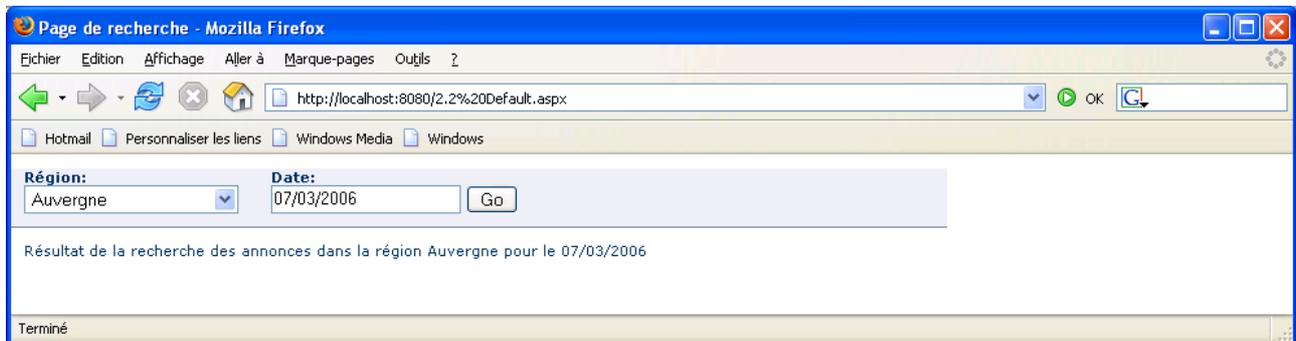
Objectif

L'objectif de cet exercice est de construire une page aspx à partir d'un formulaire HTML standard et de mettre en évidence le principe du post back avec ASP.NET 2.0.

Contexte fonctionnel

L'idée est de construire un formulaire de recherche des petites annonces à partir de deux critères : la région de publication de l'annonce représentée sous la forme d'une liste déroulante et d'une date sous la forme d'une zone de texte.

Le résultat de la recherche consiste dans un premier temps en l'affichage d'un texte résumant les critères choisis.



1.1 Partir d'une page html standard...

Déroulement de l'exercice :

1. Créez un répertoire sur le disque pour stocker la solution que vous allez développer.
2. Récupérez la feuille de style **Default.css** dans le dossier ...**Atelier 1\Fichiers Utiles** et copiez-la sur votre répertoire de projet.



La feuille de style va servir à positionner et formater les zones et les contrôles de la page.

Pour en savoir plus sur les feuilles de style, rendez-vous dans l'atelier de ce même tutorial qui traite des standards du web ☺.

3. Ouvrez le bloc-notes de Windows et enregistrez le nouveau document vide sous le nom **Default.htm** dans votre répertoire de projet.
4. Saisissez le contenu statique de la page suivante :

```
<html>
<head>
  <title>Page de recherche</title>
  <link href="Default.css" rel="stylesheet" type="text/css" />
</head>
<body>
```

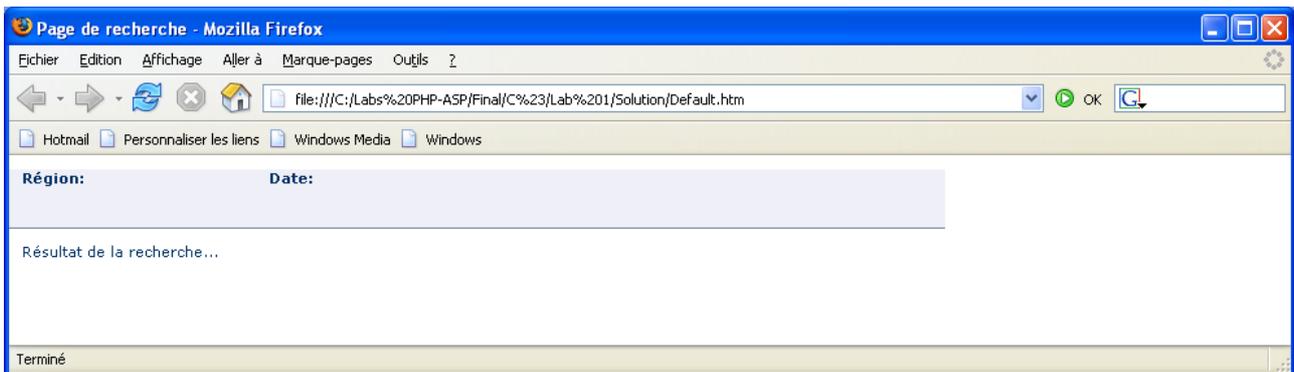
```
<div class="recherche">
  <div class="region">
    <h3>Région: </h3>
  </div>
  <div class="date">
    <h3>Date: </h3>
  </div>
</div>
<div class="resultat">
  Résultat de la recherche...
</div>
</body>
</html>
```

Quelques commentaires sur cette première page simple :

- la balise **<link>** dans l'en-tête fait référence à la feuille de style **Default.css**. Chaque balise **<div>** permet ensuite de formater la zone qu'elle englobe en faisant référence à une classe de la feuille de style via l'attribut **class**.

5. Enregistrez la page **Default.htm**.

6. Ouvrez la page dans votre navigateur. Vous obtenez :



La page est structurée en deux parties :

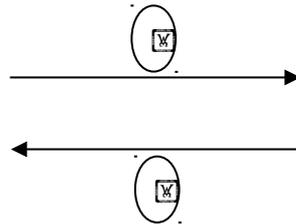
- une première zone proposant :
 - o un critère de recherche intitulé « région » qui doit être une liste déroulante listant les régions dans lesquelles sont publiées les annonces.
 - o un critère de recherche intitulé « date » qui doit être une zone de saisie où l'utilisateur entre la date à partir de laquelle il souhaite consulter les annonces.
 - o Le formulaire devra contenir également un bouton qui lancera la recherche des informations à partir de la base de données des annonces.
- une seconde zone présentant les résultats de la recherche. Pour l'instant cette zone affiche « Résultat de la recherche... ».

Note :

1.2 ...et construire un formulaire dynamique ASP.NET

Ce qui nous intéresse est bien évidemment de faire de cette page un formulaire interactif :

L'utilisateur doit pouvoir sélectionner une région à partir d'une liste déroulante, saisir une date dans une zone de texte puis cliquer sur un bouton de soumission (submit) pour envoyer les informations au serveur.



Le serveur lit les critères sélectionnés à partir des informations qu'il reçoit, exécute la recherche des annonces (typiquement à partir d'une base de données) et renvoie une page de réponse à l'utilisateur contenant la liste des annonces trouvées.

Un formulaire peut être envoyé à une adresse email, ou à un script de page tel que PHP ou ASP. Voyons comment envoyer ce formulaire de façon à ce qu'il soit pris en charge par ASP.NET sur le serveur web...

Déroulement de l'exercice :

1. Renommez la page Default.htm en **Default.aspx**.
2. Ajoutez une balise **<form>** à la page :

```
<html>
<head>
  <title>Page de recherche</title>
  <link href="Default.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form method="post" action="Default.aspx">
    <div class="recherche">
      ...
    </div>
    <div class="resultat">
      ...
    </div>
  </form>
</body>
</html>
```



Cette balise **<form>** délimite la zone dynamique proprement dite du formulaire, contenant notamment les zones de saisie de l'utilisateur dont les données devront être envoyées sur le serveur. Elle possède des attributs obligatoires :

- **method** pour indiquer de quelle manière sont envoyées les données sur le serveur : soit codées dans l'url de la requête pour la méthode « GET » soit stockées dans le corps de la requête pour la méthode « POST ». Par défaut, une page aspx utilise la méthode **POST**.
- **action** pour indiquer l'url à laquelle sont envoyées les informations. Il faut savoir qu'une page aspx poste ses données sur elle-même c'est-à-dire que l'url d'envoi est celle de **la page elle-même** tout simplement !

3. Ajoutez maintenant les contrôles de sélection/saisie et un bouton de soumission :

```
<html>
<head>
  <title>Page de recherche</title>
  <link href="Default.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form method="post" action="Default.aspx">
    <div class="recherche">
      <div class="region">
        <h3>Région: </h3>

```

```

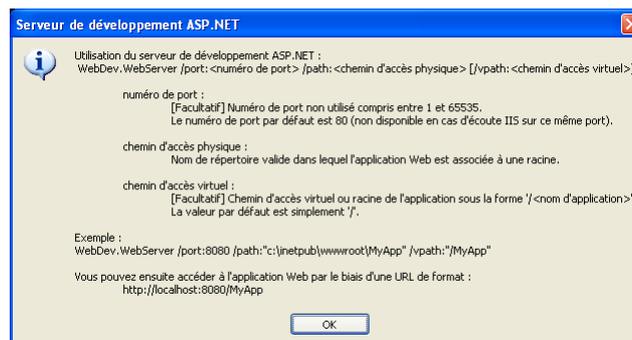
<SELECT name="ddlRegion">
  <OPTION VALUE="1">Alsace</OPTION>
  <OPTION VALUE="2">Aquitaine</OPTION>
  <OPTION VALUE="3">Auvergne</OPTION>
  <OPTION VALUE="4">Basse Normandie</OPTION>
  <OPTION VALUE="5">Bourgogne</OPTION>
  <OPTION VALUE="6">Bretagne</OPTION>
  <OPTION VALUE="7">Centre</OPTION>
  <OPTION VALUE="8">Champagne</OPTION>
  <OPTION VALUE="9">Franche Comte</OPTION>
  <OPTION VALUE="10">Languedoc Roussillon</OPTION>
  <OPTION VALUE="11">Limousin</OPTION>
  <OPTION VALUE="12">Lorraine</OPTION>
  <OPTION VALUE="13">Midi Pyrénées</OPTION>
  <OPTION VALUE="14">Nord</OPTION>
  <OPTION VALUE="15">Pays de Loire</OPTION>
  <OPTION VALUE="16">Picardie</OPTION>
  <OPTION VALUE="17">Poitou Charente</OPTION>
  <OPTION VALUE="18">Provence Côte D'Azur</OPTION>
  <OPTION VALUE="19">Ile de France</OPTION>
  <OPTION VALUE="20">Rhone Alpes</OPTION>
  <OPTION VALUE="21">Normandie</OPTION>
</SELECT>
</div>
<div class="date">
  <h3>Date: </h3>
  <input name="txtDate" type="text"/>
  <input name="btnGo" type="submit" value="Go" />
</div>
</div>
<div class="resultat">
  Résultat de la recherche...
</div>
</form>
</body>
</html>

```

4. Enregistrez la page.
5. Exécutez la page avec le serveur de test du Framework 2.0 (idéal pour se passer d'Internet Information Server pendant la phase de développement ☺) :
 - Retrouvez le serveur de test **WebDev.WebServer.exe** dans le dossier : **C:\WINDOWS\MICROSOFT.NET\Framework\v2.0.50727**
 - Exécutez-le en ligne de commande comme suit :



Pour connaître les paramètres de cette commande, il suffit de taper **WebDev.WebServer.exe** suivi de la touche **Entrer** et vous obtenez :

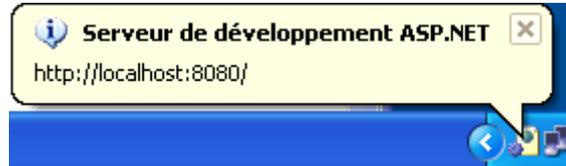


Voici la commande à saisir :

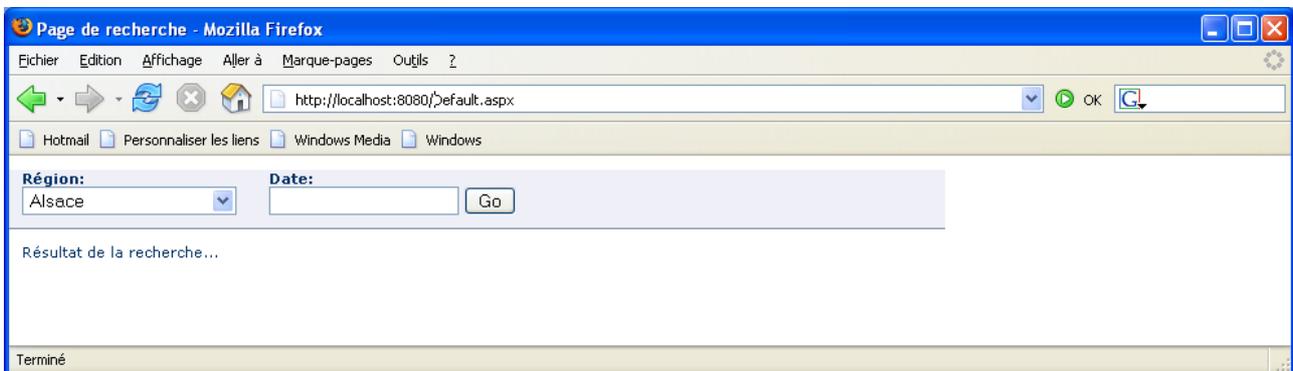
```
WebDev.WebServer.exe /port:8080 /path:"c:\VotreDossierDeProjetPourCeLab\"
```



Le serveur web de test se lance et une icône apparaît dans la zone de notification de Windows. Vous pourrez utiliser l'icône pour vérifier que votre serveur de test est bien en exécution lorsque vous exécuterez vos tests.

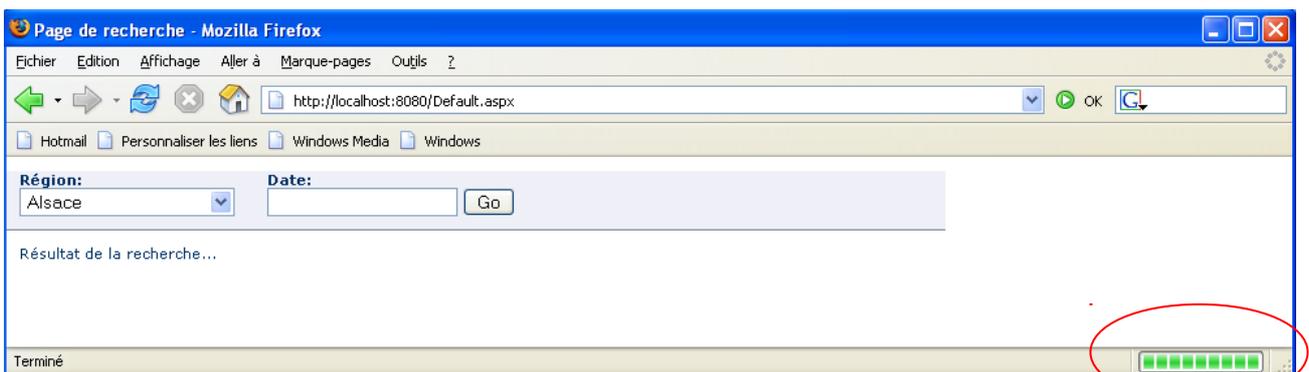


- Ouvrez le navigateur et saisissez l'adresse de la page en reprenant le numéro de port utilisé par le serveur de test (dans l'exemple 8080) : **http://localhost:8080/Default.aspx**



Vous observez maintenant la liste déroulante avec la liste des régions, la zone de texte pour la date et le bouton Go.

Remarquez qu'un clic sur le bouton de soumission déclenche un aller retour sur le serveur web, bien visible par un flash de la page et par l'apparition d'une barre de progression dans la barre de statut du navigateur pendant le temps que dure le post back. Comme aucun traitement serveur n'est encore défini dans la page, la réponse est la page elle-même identique à celle que vous aviez avant l'appel.



6. Maintenant vous allez ajouter un traitement à la page qui sera pris en charge côté serveur par ASP.NET 2.0. Comme la page est postée sur elle-même, ce traitement est directement programmé dans la page elle-même. Le mécanisme le plus simple consiste à inclure les instructions de la page qui doivent être exécutées sur le serveur à l'intérieur d'une balise server `<% %>`:

```
<html>
<head>...</head>
```

ASP.NET repère le code de traitement server à exécuter via cette balise.

```
<%
%>
<body>
<form method="post" action="Default.aspx"> ...</form>
</body>
</html>
```

Pour l'instant, nous allons simplifier le traitement de recherche des annonces et nous contenter d'afficher dans la zone de résultat délimitée par la balise **<div class=resultat>**, un résumé des critères de sélection saisis par l'utilisateur. Comment ASP.NET 2.0 permet-il de faire cela ?

L'idée de base est de vous simplifier le travail, notamment en vous épargnant l'écriture brute du texte de résultat en html, tel qu'il doit apparaître dans la page de réponse. Au lieu de cela, ASP.NET 2.0 vous propose de coder avec un langage compilé, C# ou VB.Net entre autres, en utilisant un modèle de « contrôles » avec des propriétés, des méthodes et des événements pour manipuler les différents éléments html de la page.

- Ajoutez l'attribut **runat** avec la valeur **server** aux éléments html **<SELECT>**, **<INPUT>** de type « text » et à la balise **<DIV>** de résultat et donnez-leur à chacun un **identifiant unique** (via l'attribut **id**).
- Ajoutez également **runat="server"** sur le formulaire délimité par **<FORM>**. Vous obtenez :

```
...
<form method="post" action="Default.aspx" runat="server">
  <div class="recherche">
    <div class="region">
      <h3>Région: </h3>
      <SELECT name="ddlRegion" runat="server" id="ddlRegion">
        <OPTION VALUE="1">Alsace</OPTION>
        <OPTION VALUE="2">Aquitaine</OPTION>
        <OPTION VALUE="3">Auvergne</OPTION>
        <OPTION VALUE="4">Basse Normandie</OPTION>
        <OPTION VALUE="5">Bourgogne</OPTION>
        <OPTION VALUE="6">Bretagne</OPTION>
        <OPTION VALUE="7">Centre</OPTION>
        <OPTION VALUE="8">Champagne</OPTION>
        <OPTION VALUE="9">Franche Comte</OPTION>
        <OPTION VALUE="10">Languedoc Roussillon</OPTION>
        <OPTION VALUE="11">Limousin</OPTION>
        <OPTION VALUE="12">Lorraine</OPTION>
        <OPTION VALUE="13">Midi Pyrénées</OPTION>
        <OPTION VALUE="14">Nord</OPTION>
        <OPTION VALUE="15">Pays de Loire</OPTION>
        <OPTION VALUE="16">Picardie</OPTION>
        <OPTION VALUE="17">Poitou Charente</OPTION>
        <OPTION VALUE="18">Provence Côte D'Azur</OPTION>
        <OPTION VALUE="19">Ile de France</OPTION>
        <OPTION VALUE="20">Rhone Alpes</OPTION>
        <OPTION VALUE="21">Normandie</OPTION>
      </SELECT>
    </div>
    <div class="date">
      <h3>Date: </h3>
      <input name="txtDate" type="text" runat="server" id="txtDate"/>
      <input name="btnGo" type="submit" value="Go" />
    </div>
  </div>
  <div class="resultat" runat="server" id="resultat">
    Résultat de la recherche...
  </div>
```

```
</form>
```

```
...
```



L'attribut **runat="server"** est fondamental car il indique au moteur ASP.NET qui traite la requête sur le serveur que les balises HTML, normalement vues comme de simple textes littéraux, vont pouvoir être considérées comme des contrôles accessibles par programmation via leur **id** unique.

Qu'est-ce que cela signifie ?

L'idée est que lorsque le serveur démarre le traitement de la page, un objet **Page** est chargé en mémoire et réalise une succession d'étapes dont la toute première est de créer une instance de tous les contrôles marqués avec **runat="server"**. Ces instances s'appuient sur des classes définies dans le Framework 2.0.

Pour chaque balise html marquée, on obtient donc un objet :

- basé sur une classe du Framework,
- ayant une représentation graphique,
- doté de propriétés qui le caractérisent et le définissent,
- doté de méthodes qui permettent de jouer sur son comportement,
- et doté d'évènements qui sont déclenchés en général par des actions de l'utilisateur sur le contrôle.

Ainsi un contrôle `<INPUT>` de type « submit » tel que le bouton **Go**, est chargé en mémoire par ASP.NET sur la base d'une classe nommée **HtmlButton**. Cet objet est alors accessible par programmation et comprend entre autres :

- une représentation graphique qui est celle d'une balise INPUT de type « submit »,
- une propriété **InnerText** qui permet d'inscrire un texte sur le bouton,
- une méthode **Focus()**, qui permet de positionner le focus sur le contrôle,
- un évènement **ServerClick** qui est déclenché lorsque l'utilisateur clique sur le bouton.



Le formulaire doit être également **runat="server"** et est alors vu comme un contrôle de type **HtmlForm**. Il doit lui-même contenir tous les contrôles que vous voulez poster sur le serveur.

A noter que :

- sa propriété **method** est « POST » par défaut,
- et que sa propriété **action** est l'url de la page elle-même et ne peut être modifiée. Bien que vous ne puissiez pas poster la page sur une adresse différente qu'elle-même, il est possible en ASP.NET de réaliser un post back sur une autre page via un mécanisme simple décrit dans l'atelier 11 du coach ASP.NET, rubrique « Cross-Page Posting ».

- Avant de programmer le traitement serveur, il reste à indiquer le langage de votre choix dans ce qu'on appelle une « directive » de page. Celle-ci est écrite au tout début de la page et est lue côté serveur par ASP.NET. Elle permet de configurer de manière très précise votre page web via un ensemble d'attributs qui renseignent le serveur sur la façon dont il doit exécuter la page.

Dans le cas présent, nous allons indiquer au serveur le langage de votre choix, en utilisant l'attribut **Language** :

Code VB.NET

```
<%@ Page Language="VB" %>
<html>
...
</html>
```

Code C#

```
<%@ Page Language="C#" %>
<html>
...
</html>
```

- Puisque vous avez accès aux éléments <SELECT>, <INPUT> et <DIV> via les contrôles server chargés en mémoire par ASP.NET 2.0, il ne vous reste plus qu'à programmer la lecture de la date dans la zone de texte et l'écriture du texte final dans la zone de résultat :

Code VB.NET

```
<%  
If Not txtDate.Value = "" Then  
    resultat.InnerHtml = "Résultat de la recherche pour le " & txtDate.Value  
End If  
%>
```

Code C#

```
<%  
if (txtDate.Value != "")  
{  
    resultat.InnerHtml = "Résultat de la recherche pour le " + txtDate.Value;  
}  
%>
```



Ces lignes d'instruction utilisent :

- la propriété **Value** du contrôle **txtDate** de type **HtmlInputText** pour lire la valeur de la date saisie dans la zone de texte et envoyée au serveur lors d'un post back.
- la propriété **InnerHtml** pour modifier le contenu html du contrôle **resultat** de type **HtmlGenericControl** qui représente la balise <DIV> d'id resultat.

Vous constatez que, grâce à ces contrôles chargés par le serveur, il est facile de lire leur valeur et également d'écrire et modifier celle-ci. Cela signifie qu'il n'est pas nécessaire de construire une nouvelle page contenant la réponse html à renvoyer au navigateur client. La modification est faite directement par le serveur dans la page elle-même. C'est donc la même page qui redescend sur le navigateur avec les modifications réalisées !

Pourquoi faire un test de la date ? Tout simplement parce que le serveur est susceptible d'exécuter plusieurs fois ce traitement, au moins une première fois lors du tout premier chargement de la page, puis autant de fois que l'utilisateur provoque un post back sur la page via le bouton Go. Or au tout premier chargement, l'utilisateur n'a pas encore saisi de date puisqu'il n'a pas encore vu la page...

Autre exemple de traitement serveur : on pourrait par exemple écrire dans la zone de texte pour l'initialiser au chargement de la page avec la date du jour. Cela donnerait :

Code VB.NET

```
<%  
txtDate.Value = DateTime.Now.ToShortDateString()  
%>
```

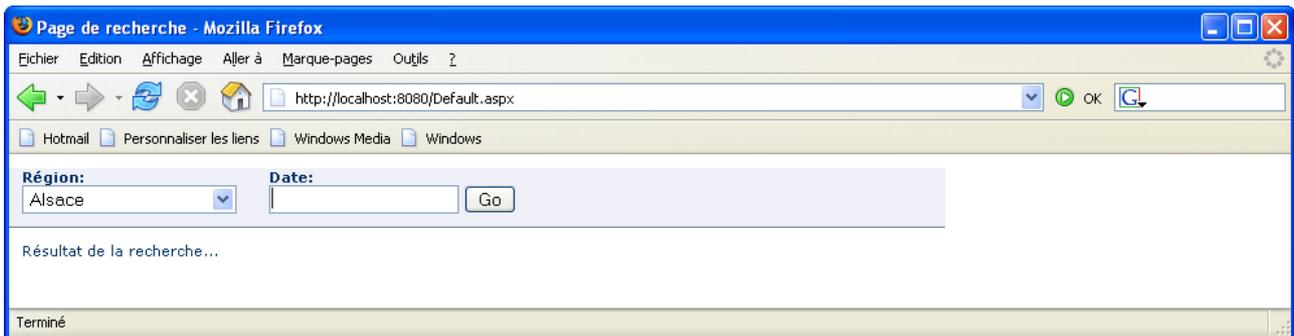
Code C#

```
<%  
txtDate.Value = DateTime.Now.ToShortDateString() ;  
%>
```

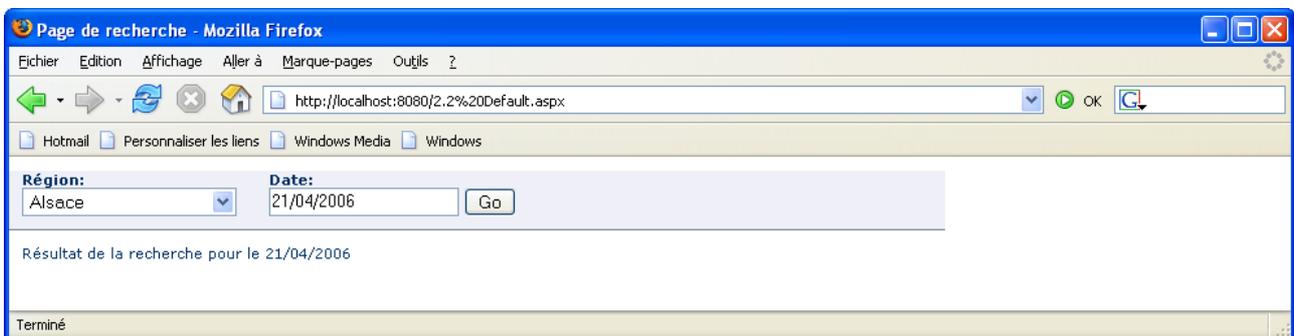


La propriété **DateTime.Now** du Framework 2.0 donne la date du jour que l'on formate avec **ToShortDateString()**.

- Enregistrez puis exécutez la page.
A la première requête sur la page, la zone de texte **txtDate** ne contient pas encore de donnée donc le test que vous réalisez en début de code retourne FAUX. Du coup le texte de la balise <DIV> de resultat reste tel quel en html, c'est-à-dire « **Résultat de la recherche...** » :



- Une fois la page de réponse affichée dans le navigateur, saisissez une date dans la zone de texte.
- Cliquez sur le bouton **Go** pour provoquer un post back sur le serveur. C'est la deuxième fois que le serveur exécute la page et cette fois-ci, le test renvoie VRAI, donc la zone de résultat est modifiée en conséquence. La page redescend sur le navigateur.



Notez qu'avec ce principe, vous pouvez donc passer N fois dans le même traitement d'une page, une fois au chargement de la page lors de la requête initiale puis autant de fois que de post back initié sur la page, et ce pour chaque utilisateur effectuant une requête.

- Ajoutez à la page, en suivant la même logique, la lecture de la sélection dans la liste déroulante de façon à compléter le texte de résultat comme ceci :

Code VB.NET

```
<%
If Not txtDate.Value = "" Then
    resultat.InnerHtml = "Résultat de la recherche des annonces dans la région " _
        & ddlRegion.Items(ddlRegion.SelectedIndex).Text & " pour le " + txtDate.Value
End If
%>
```

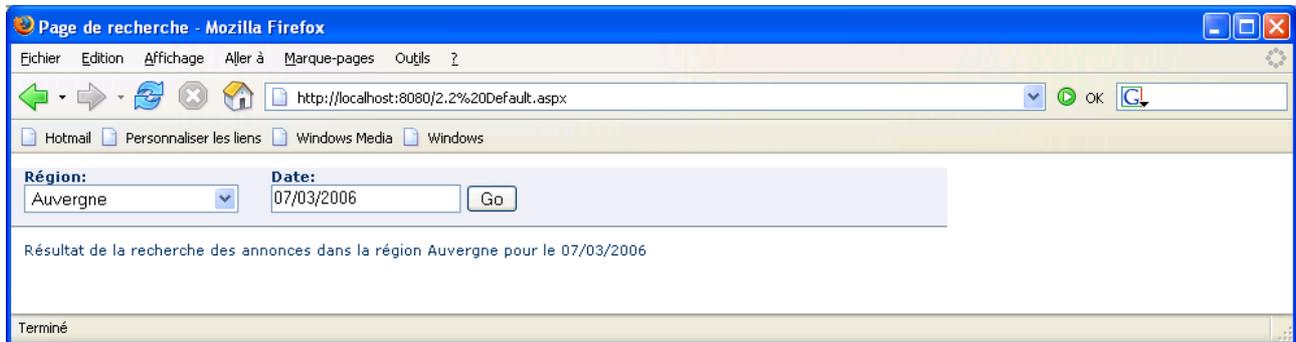
Code C#

```
<%
if (txtDate.Value != "")
{
    resultat.InnerHtml = "Résultat de la recherche des annonces dans la région "
    + ddlRegion.Items[ddlRegion.SelectedIndex].Text + " pour le " + txtDate.Value;
}
%>
```



ddlRegion est un contrôle de type **HtmlSelect** et contient une collection d'options appelée **Items**. Chacune des options est accessible via un index, l'élément sélectionné en cours étant positionné à l'index donné par la propriété **SelectedIndex** du contrôle.

- Enregistrez puis exécutez la page.
- Sélectionnez une région dans la liste, entrez une date dans la zone de texte et cliquez sur **Go**.



Bravo ! Vous avez réalisé votre première page dynamique en ASP.NET !

Note :

1.3 Utiliser les contrôles Html Serveur

Vous l'avez compris, tous les contrôles Html standards sont représentés dans le Framework 2.0 par des classes du type **HtmlForm**, **HtmlInputText**, **HtmlInputButton**, ou plus générique comme **HtmlGenericControl** etc... Ces classes nous permettent d'avoir accès aux éléments Html très facilement dans un traitement côté serveur. La seule condition pour les utiliser est de donner un **id** à vos balises et de les configurer avec **runat="server"**.

Vous pouvez retrouver d'un simple clic la définition de tous ces contrôles sur la bibliothèque MSDN en ligne à l'adresse suivante : <http://msdn2.microsoft.com/en-us/library/620b4zf.aspx>

Supposons qu'au lieu d'écrire un simple texte de résultat, on veuille construire un tableau html avec plusieurs annonces issues de la recherche. Toujours avec cette même approche de contrôles serveur, cette opération peut se faire très simplement grâce à la classe **HtmlTable** qui représente le tableau html standard et qui contient des éléments de type **HtmlTableRow** (ligne) eux-mêmes constitués de **HtmlTableCell** (cellule).

Déroulement de l'exercice :

1. Ajoutez un tableau vide dans la zone de résultat de la page à la place du texte initial :

```
<form method="post" action="Default.aspx" runat="server">
...
<div class="resultat" runat="server" id="resultat">
  <table id="tblResultat" class="gridcontent" runat="server">
    </table>
</div>
</form>
```

Ne pas oublier l'attribut runat !

Format défini dans la feuille de style

2. Mettez en commentaire le code de l'exercice précédent dans la balise <%%>.

Code VB.NET

```
<%  
'If Not txtDate.Value = "" Then  
'    resultat.InnerHtml = "Résultat de la recherche des annonces dans la région " _  
'    & ddlRegion.Items(ddlRegion.SelectedIndex).Text & " pour le " + txtDate.Value  
'End If  
%>
```

Code C#

```
<%  
/*if (txtDate.Value != "")  
{  
    resultat.InnerHtml = "Résultat de la recherche des annonces dans la région "  
    + ddlRegion.Items[ddlRegion.SelectedIndex].Text + " pour le " + txtDate.Value;  
}*/  
%>
```

3. Ajoutez le code suivant affichant une seule annonce (codée en dur pour simplifier) dans le tableau de résultat lorsque la région sélectionnée est l'**Auvergne** (indexe = 2) :

Code VB.NET

```
<%  
'..'  
If ddlRegion.SelectedIndex = 2 Then  
  
    ' Créer une nouvelle ligne de tableau  
    Dim ligne As HtmlTableRow = New HtmlTableRow()  
  
    ' Créer une nouvelle cellule de tableau pour afficher l'email  
    Dim celluleEmail As HtmlTableCell = New HtmlTableCell()  
    ' Afficher dans la cellule un texte littéral en utilisant le control LiteralControl  
    celluleEmail.Controls.Add(New LiteralControl("gege@hotmail.fr"))  
    'Ajouter la cellule à la collection de cellules de la ligne  
    ligne.Cells.Add(celluleEmail)  
    'Recommencer pour afficher la ville et le code postal de l'annonce  
    Dim celluleVille As HtmlTableCell = New HtmlTableCell()  
    celluleVille.Controls.Add(New LiteralControl("Paris"))  
    ligne.Cells.Add(celluleVille)  
    Dim celluleCodePostal As HtmlTableCell = New HtmlTableCell()  
    celluleCodePostal.Controls.Add(New LiteralControl("75000"))  
    ligne.Cells.Add(celluleCodePostal)  
  
    'Ajouter la ligne à la collection de lignes du tableau  
    tblResultat.Rows.Add(ligne)  
End If  
%>
```

Code C#

```
<%  
/*...*/  
if (ddlRegion.SelectedIndex == 2)  
{  
    // Créer une nouvelle ligne de tableau  
    HtmlTableRow ligne = new HtmlTableRow();  
  
    // Créer une nouvelle cellule de tableau pour afficher l'email  
    HtmlTableCell celluleEmail = new HtmlTableCell();
```

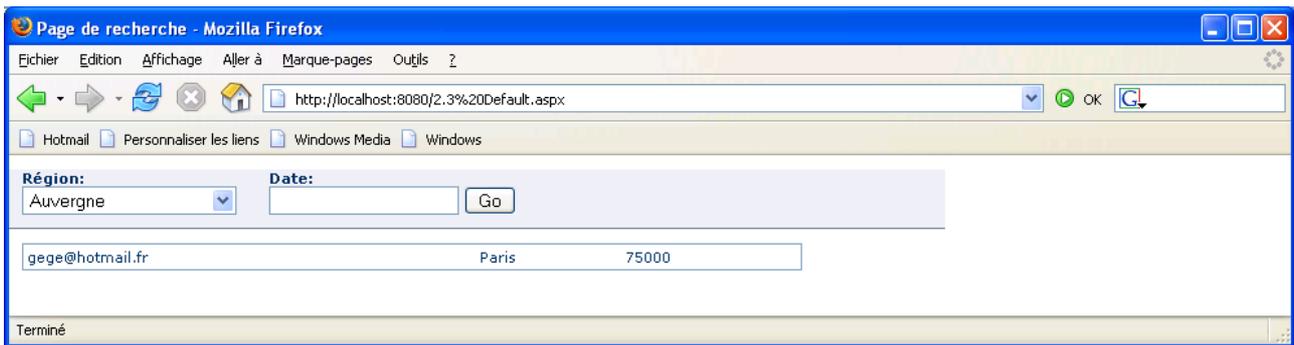
```
// Afficher dans la cellule un texte littéral en utilisant le control LiteralControl
celluleEmail.Controls.Add(new LiteralControl("gege@hotmail.fr"));
//Ajouter la cellule à la collection de cellules de la ligne
ligne.Cells.Add(celluleEmail);
//Recommencer pour afficher la ville et le code postal de l'annonce
HtmlTableCell celluleVille = new HtmlTableCell();
celluleVille.Controls.Add(new LiteralControl("Paris"));
ligne.Cells.Add(celluleVille);
HtmlTableCell celluleCodePostal = new HtmlTableCell();
celluleCodePostal.Controls.Add(new LiteralControl("75000"));
ligne.Cells.Add(celluleCodePostal);

//Ajouter la ligne à la collection de lignes du tableau
tblResultat.Rows.Add(ligne);
}
%>
```



Notez que **tblResultat** est directement accessible depuis le code serveur puisque vous avez défini la table html avec l'attribut **runat="server"**. C'est un contrôle de type **HtmlTable** contenant une collection de lignes (Rows). Remarquez également la méthode **Add()** qui permet d'ajouter un élément à une collection. A l'inverse la méthode **Remove()** permet de supprimer un élément de la collection.

- Enregistrez puis exécutez la page.
- Sélectionnez l'**Auvergne** dans la liste puis cliquez sur **Go**.



Et voilà ! Vous avez généré un tableau Html en dynamique en utilisant les contrôles Html serveur d'ASP.NET 2.0 !

- Faites un clic droit sur la page pour afficher le code source.



Vous constatez que :

- l'attribut **runat** n'apparaît pas dans la page de réponse retournée par le serveur web. En effet, à la fin de l'exécution de la page sur le serveur, chacun des contrôles html serveur donnent à ASP.NET sa « représentation » (son rendu) html standard pour que celle-ci soit intégrée à la page de réponse que vous voyez.
- la table **tblResultat** qui a été générée est bien un tableau html standard, correspondant au rendu du contrôle **HtmlTable** une fois construit.
- la page contient un champ caché (type hidden) nommé **__VIEWSTATE**. A quoi sert-il ? C'est justement l'objet du prochain exercice ☺...

```

2.3%20Default[1] - Bloc-notes
Fichier Edition Format Affichage ?

<html>
<head>
  <title>
    Page de recherche
  </title>
  <link href="default.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <form name="ctl100" method="post" action="2.3 Default.aspx" id="ctl100">
  <div>
  <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUKLTExMzk1MjYzM2RkErY7wZ9ozw.
  </div>

  <div class="recherche">
    <div class="region">
      <h3>Région: </h3>
      <select name="ddlRegion" id="ddlRegion">
        <option value="1">Alsace</option>
        <option value="2">Aquitaine</option>
        <option selected="selected" value="3">Auvergne</option>
        <option value="4">Basse Normandie</option>
        <option value="5">Bourgogne</option>
        <option value="6">Bretagne</option>
        <option value="7">Centre</option>
        <option value="8">Champagne</option>
        <option value="9">Franche Comte</option>
        <option value="10">Languedoc Roussillon</option>
        <option value="11">Limousin</option>
        <option value="12">Lorraine</option>
        <option value="13">Midi Pyr&#233;n&#233;es</option>
        <option value="14">Nord</option>
        <option value="15">Pays de Loire</option>
        <option value="16">Picardie</option>
        <option value="17">Poitou Charente</option>
        <option value="18">Provence C&#244;te D'Azur</option>
        <option value="19">Ile de France</option>
        <option value="20">Rhone Alpes</option>
        <option value="21">Normandie</option>
      </select>
    </div>
    <div class="date">
      <h3>Date: </h3>
      <input name="txtDate" type="text" id="txtDate" />
      <input name="btnGo" type="submit" value="Go" />
    </div>
  </div>
  <div id="resultat" class="resultat">
    <table id="tblResultat" class="gridcontent">
      <tr>
        <td>gege@hotmail.fr</td>
        <td>Paris</td>
        <td>75000</td>
      </tr>
    </table>
  </div>
  <div>
    <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="/wEWAwLD4qLrDA
  </div></form>
</body>
</html>

```

Note :

