

TP 1 : WebForms

Objectif : Créer une page ASP.NET avec Visual Studio et d'utiliser son interface en environnement Web.

Définition : WebForm

Les WebForms sont des composants actifs proposés par l'environnement ASP.NET que l'on peut ajouter à un environnement de conception de page Web de manière graphique, comme si l'on concevait une interface client lourd VB ou C#. Ces éléments peuvent être classés de la manière suivante :

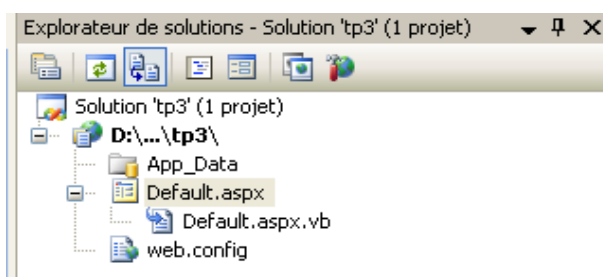
- Les éléments classiques que l'on peut trouver dans un formulaire HTML (bouton, label, zone de texte, combo box, radio, check...)
- Les composants utiles, mais plus complexes à créer dans le monde Web (calendrier graphique HTML, bannière publicitaire dynamique)
- Les outils de « contrainte » permettant de valider les informations saisies par l'utilisateur.

Le développement de sites Web basés sur les WebForms s'approche davantage de la programmation d'interfaces graphiques typiques (Formulaires Windows Form par exemple) que de la programmation Web classique (traitement d'une requête, génération de la prochaine page).

Les WebForms sont des composants exécutés côté serveur Web (IIS en l'occurrence), et génèrent des pages 100% standard contenant à la fois HTML et JavaScript. Cela va même plus loin : les WebForms détectent le type de navigateur qui se connecte, et génèrent une page (HTML + JavaScript) compréhensible par ce navigateur en procédant aux adaptations nécessaires (balises non supportées, etc...).

Atelier 1 : Créer un site Web avec Visual Studio

1. Lancer **Visual Studio**.
2. Créez un nouveau projet de type Web :
 - Cliquez le menu **Fichier > Nouveau > Site Web...**
 - Sélectionnez le modèle de projet web **Site Web Forms ASP.Net**
 - Dans la liste **Emplacement**, sélectionnez **Système de fichiers**.
 - Sélectionnez le chemin du répertoire et donnez un nom au site : **TP1**
 - Dans la liste **Langage**, sélectionnez votre langage de prédilection.
 - Cliquez sur **OK**.



Visual Studio .NET crée pour vous le site Web ASP.NET dans le répertoire du travail, ainsi qu'une page ASPX « Default.aspx » qui est affiché en mode design. Pour basculer en mode code, cliquez sur **Source**.

Vous pouvez constater aussi la présence des fichiers **web.config** pour la configuration du site, **Default.aspx** et **Default.aspx.cs** qui contiennent respectivement le code HTML et C# de la page d'accueil qui sera chargée par défaut.

Pour ajouter un élément existant dans votre application, par exemple une image ou une page web existante, il suffit de copier le fichier correspondant dans le répertoire de l'application **tp1** depuis l'**Explorateur de Windows**.

3. Observez la page **Default.aspx** générée :

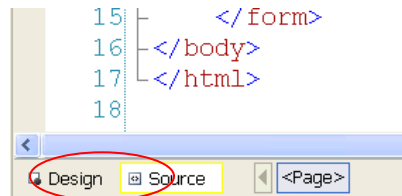
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transi:

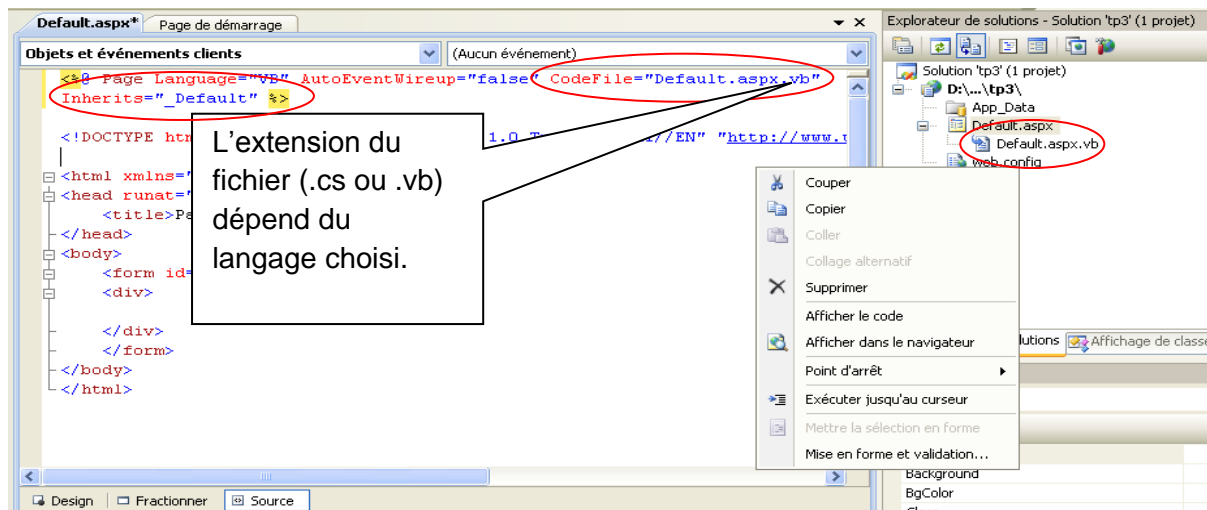
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <div>

    </div>
  </form>
</body>
</html>
```

- Basculer en mode **Design**, avec l'onglet au bas gauche de la page, pour voir la représentation graphique de la page avec l'éditeur WYSIWYG de Visual Studio. La page est vierge pour l'instant.



- Revenir en mode **Source**.
- La directive **@Page** fait référence à un autre fichier, celui contenant le code « behind » : **Default.aspx.cs (C#)** ou **Default.aspx.vb (VB.NET)**. Elle indique également le nom de la classe en code-behind : **_Default**.




4. Ouvrez le fichier **Default.aspx.cs** :

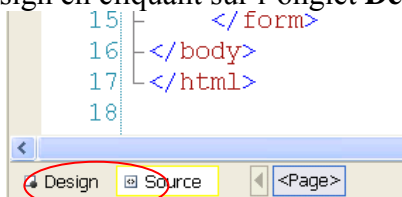
- Pour afficher le fichier de code, faites un clic droit sur la page > **Afficher le code** ou cliquez sur le fichier dans l'Explorateur de solutions.
- Observez la définition de la classe **_Default** :

C'est l'ensemble de ces fichiers qui constituera la page web complète exécutée par le runtime ASP.NET pour répondre à la requête. C'est ce principe des classes partielles qui garantit que vous pourrez invoquer les IDs de vos contrôles serveur définis dans la page de présentation depuis la page de code-behind, sans avoir à les redéfinir. Résultat : votre code est séparé de la présentation de la page, propre, lisible et donc plus facile à maintenir !

Atelier 2 : Créer une première page Web

Nous allons afficher un message de bienvenue par l'intermédiaire du contrôle server de type Label.

1. Ouvrez la page Default.aspx dans le concepteur de vue en mode Design :
 - Depuis l'**Explorateur de solutions**, cliquez sur l'icône **Concepteur de vue** .
2. Dessiner un contrôle Label sur la page :
 - Passez en mode Design en cliquant sur l'onglet **Design** en bas à gauche.



- Depuis la boîte à outils, faites un glisser déplacer d'un contrôle **Label** de la catégorie **Standard** sur la page.
3. Vous allez maintenant répondre à l'évènement de chargement de la page en mémoire de façon à afficher un message de bienvenue :
 - Afficher la page **Default.aspx** en mode **Design** et double cliquez n'importe où sur la page pour faire apparaître la fenêtre de code behind et générer automatiquement la signature de la procédure de réponse à l'évènement de chargement de la page (c'est l'évènement associé par défaut au double clic sur la page).

```
public partial class _Default : System.Web.UI.Page
{
    0 références
    protected void Page_Load(object sender, EventArgs e)
    {
    }
}
```

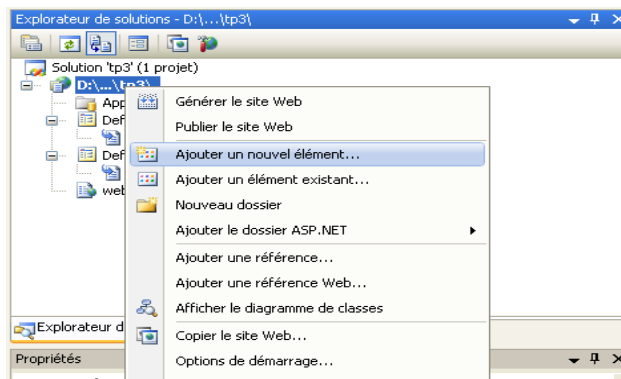
- Saisissez le code d'affichage du message dans le contrôle label :

Lable1.Text = "Bienvenue sur mon site"

4. Lancer l'application (menu **Déboguer** > **Exécuter sans débogage** ou **F5**)
5. Vous constatez que le message de bienvenue s'affiche.

Atelier 3 : Créer une deuxième page Web

1. Créez une deuxième page Default2.aspx depuis l'**Explorateur de solutions** :
 - Faites un clic droit sur le projet > **Ajouter un nouvel élément**.

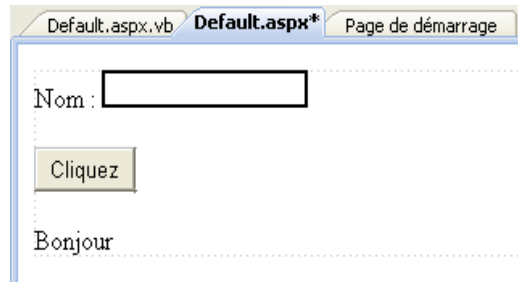


- Sélectionnez le modèle **Formulaire Web**.
 - Laissez le nom par défaut : **Default2.aspx**.
2. Dessiner des contrôles sur la page :
 - Dans le mode Design de la page Default2.aspx, ajouter les contrôles suivants sur la page Web (en faisant un glisser/déplacer des contrôles vers la page par exemple) et changer leurs propriétés en suivant le tableau ci-après :

Contrôle	Propriété	Contenu
Un "label" : Label1	Text	"Nom :"
Un "textbox" à droite de Label1 : TextBox1	BorderWidth	2
Un "button" sous Label1 : Button1	Text	"Cliquez"
Un "label" sous le bouton : Label2	Text	"Bonjour"

Remarque : dans la propriété BorderWidth, par défaut, l'unité de mesure est en "px" (pixel). Cela correspond bien aux normes HTML.

Votre page doit ressembler à ceci :



- Passer en mode HTML afin d’observer le code HTML généré par Visual Studio .NET
3. Nous allons essayer de faire en sorte que lorsque l’utilisateur clique sur le bouton, le Label2 soit mis à jour avec le contenu de la zone de texte.
- Basculer en mode **Design**
 - Double cliquer sur le bouton “*Cliquez*”
 - Le gestionnaire d’événement associé à l’événement Click du bouton est automatiquement généré dans le code serveur (code behind) par Visual Studio .NET.
- Dans votre événement "Button1_Click", tapez cette ligne :

Label2.Text = "Bonjour " + TextBox1.Text

4. Tester le fonctionnement du formulaire.
- Lancer l’application (menu **Déboguer** > **Exécuter sans débogage** ou **F5**)
 - Remplir le champ texte et cliquer sur le bouton
 - Observer le résultat produit
 - Regarder la source HTML de la page obtenue pour constater qu’il n’y a pas de référence aux lignes de code tapées précédemment.

Exercices :

Exercice 1 : Date

Créer une page qui affiche la date actuelle.

Exemple d’affichage : Nous sommes le : mercredi 20 Avril 2011

Exercice 2 : Boucle

- Créer une page *boucle.aspx* permettant d’afficher 10 fois la phrase « *Page dynamique en ASP.NET* »
- Modifier la page précédente de telle façon que la taille de la police d’une phrase s’incrémente de 1 par rapport à la phrase précédente.

Exercice 3 : Calcul du prix TTC

On désire réaliser une calculatrice de prix TTC.

Créer un formulaire permettant de calculer le montant HT et TTC d’une commande d’un produit à partir du nom de produit, prix unitaire, quantité et TVA.

Prévoir le cas où les données sont invalides.

Exercice 4 : Accès par mot de passe

Créer une page d’authentification en demandant à l’utilisateur la saisie d’un nom d’utilisateur et un mot de passe (constantes).

Si les identifiants sont corrects afficher dans un message de bienvenue, sinon afficher un message d’erreur.

Améliorer l’interface en affichant, au-dessous du formulaire, le message de succès en vert et celui d’erreur en rouge.

Exercice 5 : Accès par mot de passe

Modifier l’exercice précédent comme suit :

- Initialiser le champ login par « Tapez votre login »
- Initialiser le champ mot de passe par « Tapez votre mot de passe »

NB : la propriété *isPostBack* de la page permet de vérifier s’il s’agit d’un premier chargement de la page ou non c.à.d. un chargement après un événement déclenché par l’utilisateur.

Exercice 6 : Convertisseur de devise

Le but de cet exercice est de réaliser une application Web en ASP.NET qui permet la conversion d’un montant en **Dollars U.S** en plusieurs devises : **Euros**, **Yen Japonai** et le **Dollars Canadien**

On donne les taux suivants :

1 Dollard U.S = 0,91534 Euro

1 Dollard U.S = 119,792 Yen Japonais

1 Dollard U.S = 1,40515 Dollars Canadien

1) Réaliser une page web qui permet à l'utilisateur la conversion d'un montant en Dollars U.S saisi dans une zone de texte en devises choisies dans des cases à cocher.

Pour cela on dispose des contrôles suivants :

- Le contrôle TextBox **txtDollards** : contient la somme en dollars à convertir
- 3 contrôles CheckBox : permettent de sélectionner les devises
- Le contrôle Button **btnConvertir** : permet de déclencher la conversion
- Le contrôle Label **lblDollarsEnDevise** : affiche le résultat de la conversion en devise.

2) Modifier l'interface en changeant les cases à cocher par une liste déroulante (DropDownList) pré-remplie avec les 3 devises et en gérant les exceptions sur le montant saisi.