

# *Application Web Côté Serveur*

## *Page maitre*

## Présentation

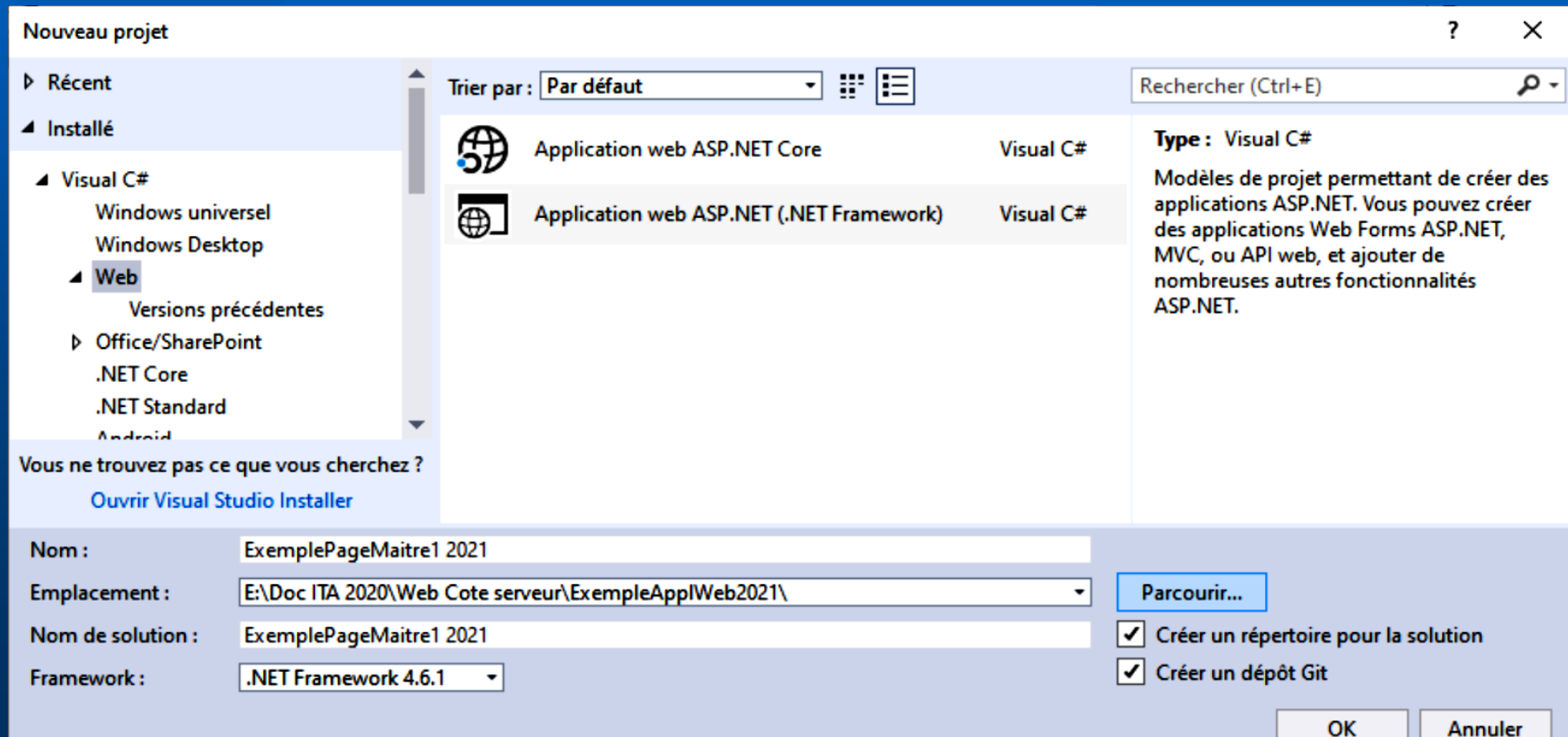
*Un page maître (master page) définit l'apparence et le comportement standard souhaitée pour toutes les pages (ou groupe de pages) de votre application.*

*Un page maitre est une page qui va contenir des éléments s'affichant de la même manière sur une partie ou sur la totalité du site, et aussi des éléments variant suivant chaque page du site (contacts, news, téléchargements...). Un page maitre va être composée de la partie commune de l'interface pour un ensemble donné de pages.*

*L'avantage de cela est que nous n'aurons plus qu'à créer un seul page maitre pour intégrer les éléments inchangés (bannière, entête, menus...) sur une étendue de pages données. De plus, si nous voulons modifier un élément de l'interface de notre site nous n'aurons qu'à le faire qu'une seule fois dans le page maitre, toutes les pages l'utilisant seront ainsi immédiatement modifiées*

# Création de page maitre


1. Lancez Visual Studio.
2. Créez un nouveau projet de type Web.





# Création de page maitre


## 3. Choisir nouvelle application vide.


Nouvelle application web ASP.NET - ExemplePageMaitre1 2021


  
Vide


  
Web Forms

  
MVC

  
Web API

  
Single Page Application

  
Application API Azure

  
Azure Mobile App

Modèle de projet vide pour créer des applications ASP.NET. Ce modèle n'a aucun contenu.  
[En savoir plus](#)

Ajoutez des dossiers et des références de base pour :

☒ Web Forms   ☐ MVC   ☐ API web

☐ Activer la prise en charge de Docker Compose (Nécessite [Docker pour Windows](#))

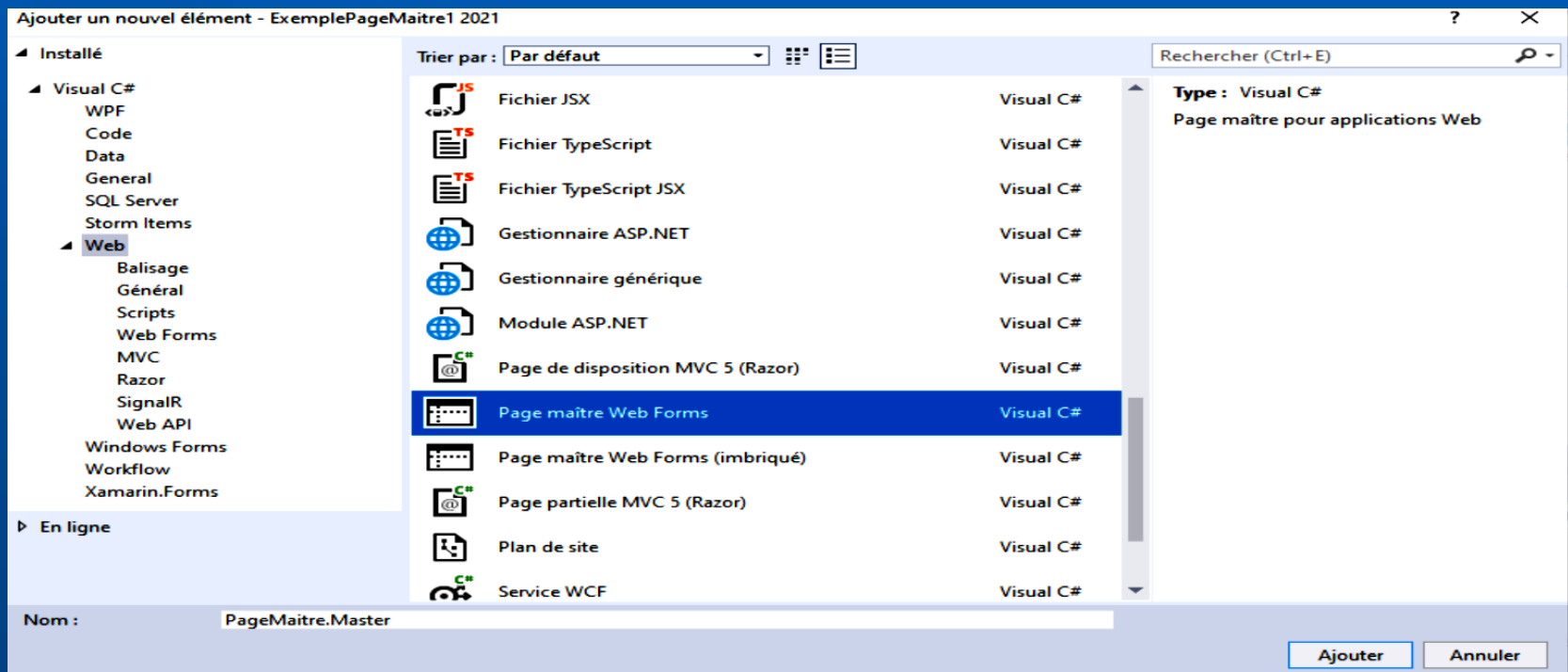
☐ Ajouter des tests unitaires

Nom du projet de test :

Authentification : **Aucune authentification**

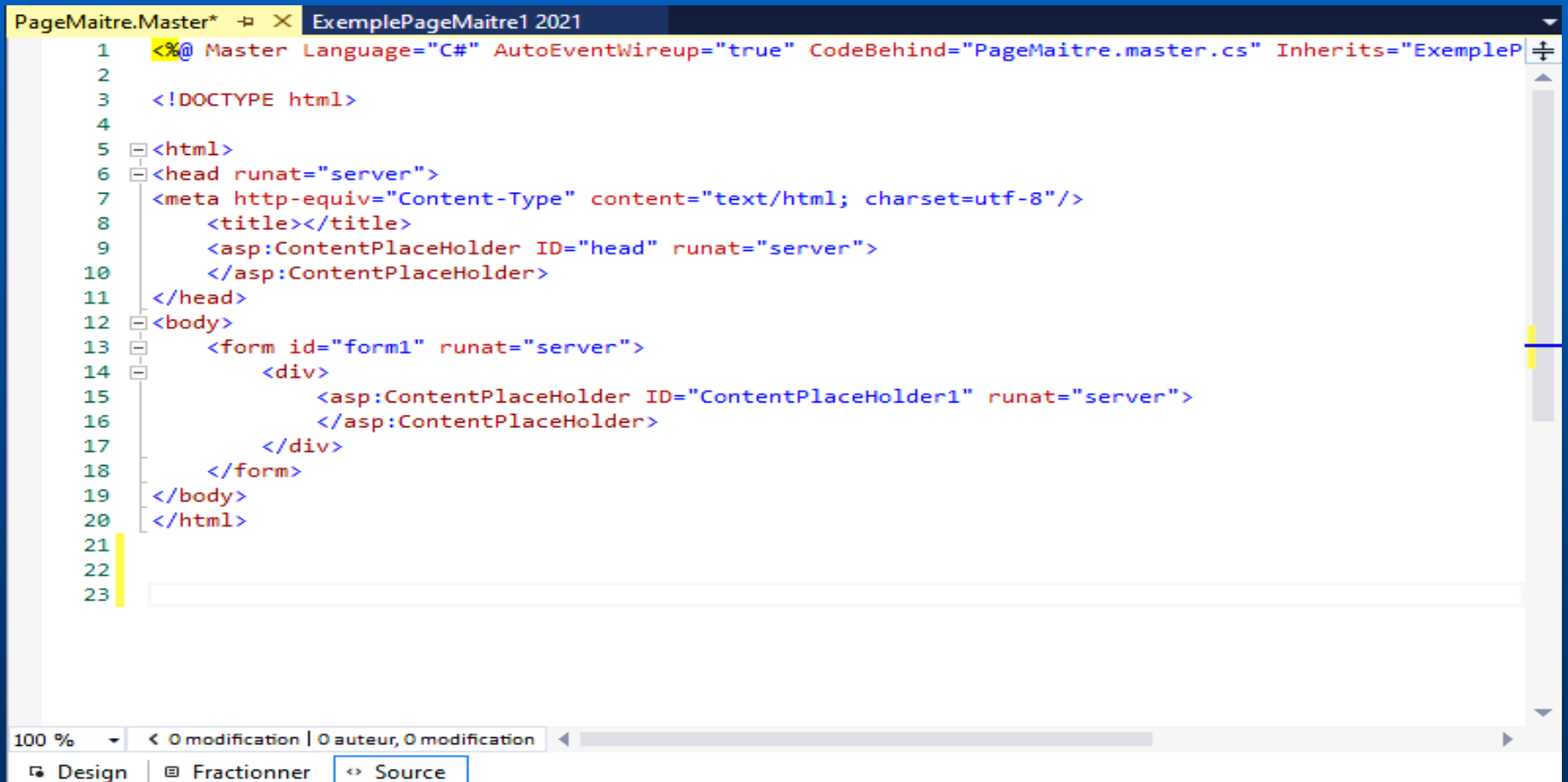
## Création de page maitre

4. cliquez avec le bouton droit puis choisissez "ajouter un nouvel élément".
5. Choisissez alors "page maitre",



*Vous pouvez remarquer qu'elle porte l'extension .master*

# Création de page maitre



```
1  <%@ Master Language="C#" AutoEventWireup="true" CodeBehind="PageMaitre.master.cs" Inherits="ExempleP
2
3  <!DOCTYPE html>
4
5  <html>
6  <head runat="server">
7      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8      <title></title>
9      <asp:ContentPlaceHolder ID="head" runat="server">
10     </asp:ContentPlaceHolder>
11 </head>
12 <body>
13     <form id="form1" runat="server">
14         <div>
15             <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
16             </asp:ContentPlaceHolder>
17         </div>
18     </form>
19 </body>
20 </html>
21
22
23
```

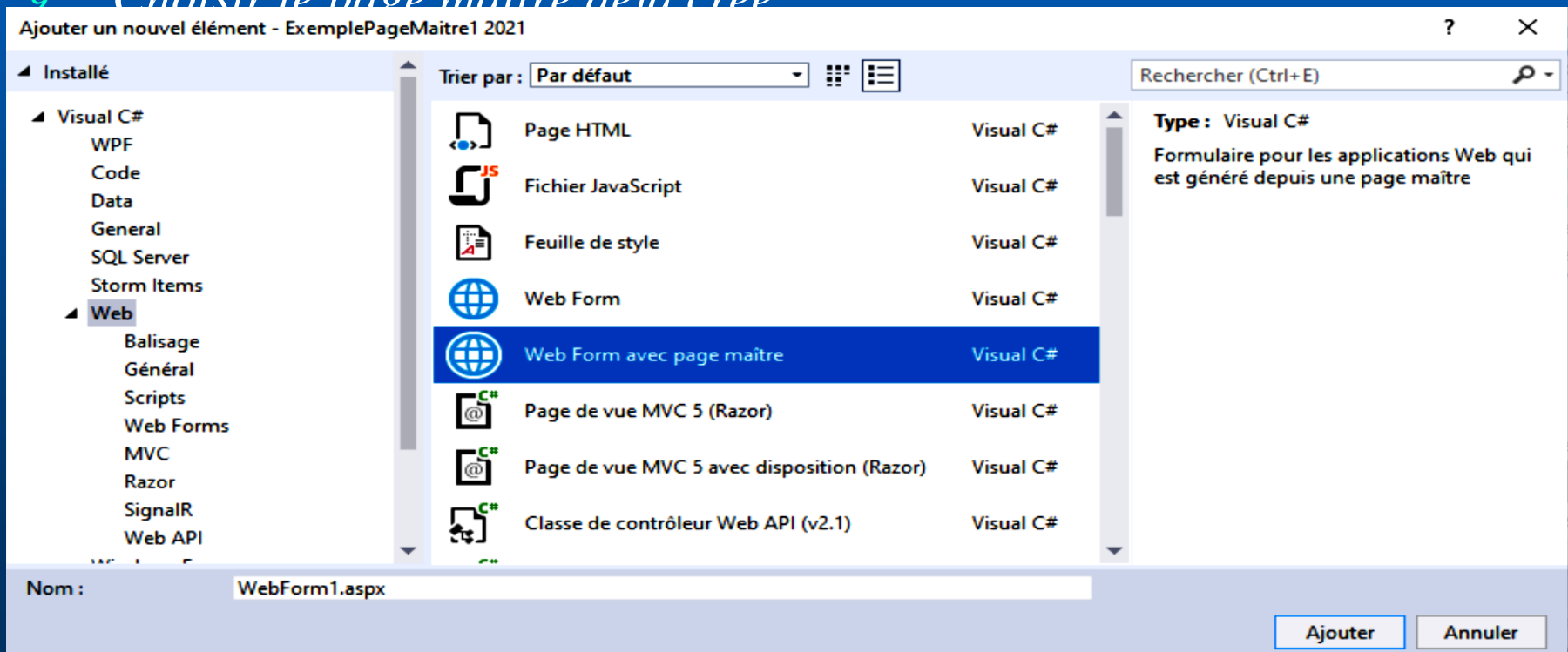
## *Création de page maitre*

- 6. Une fois la page créé, Ajouter des contrôles (un menu par exemple) dans la partie non atteinte par le contrôle ContentPlaceHolder , Placer de(s) ContentPlaceHolder là où vous souhaitez que le contenu soit modifiable.*

*Pour ajouter un ContentPlaceHolder, faire un glisser/déposer depuis la boîte à outils.*

# Création de page maitre

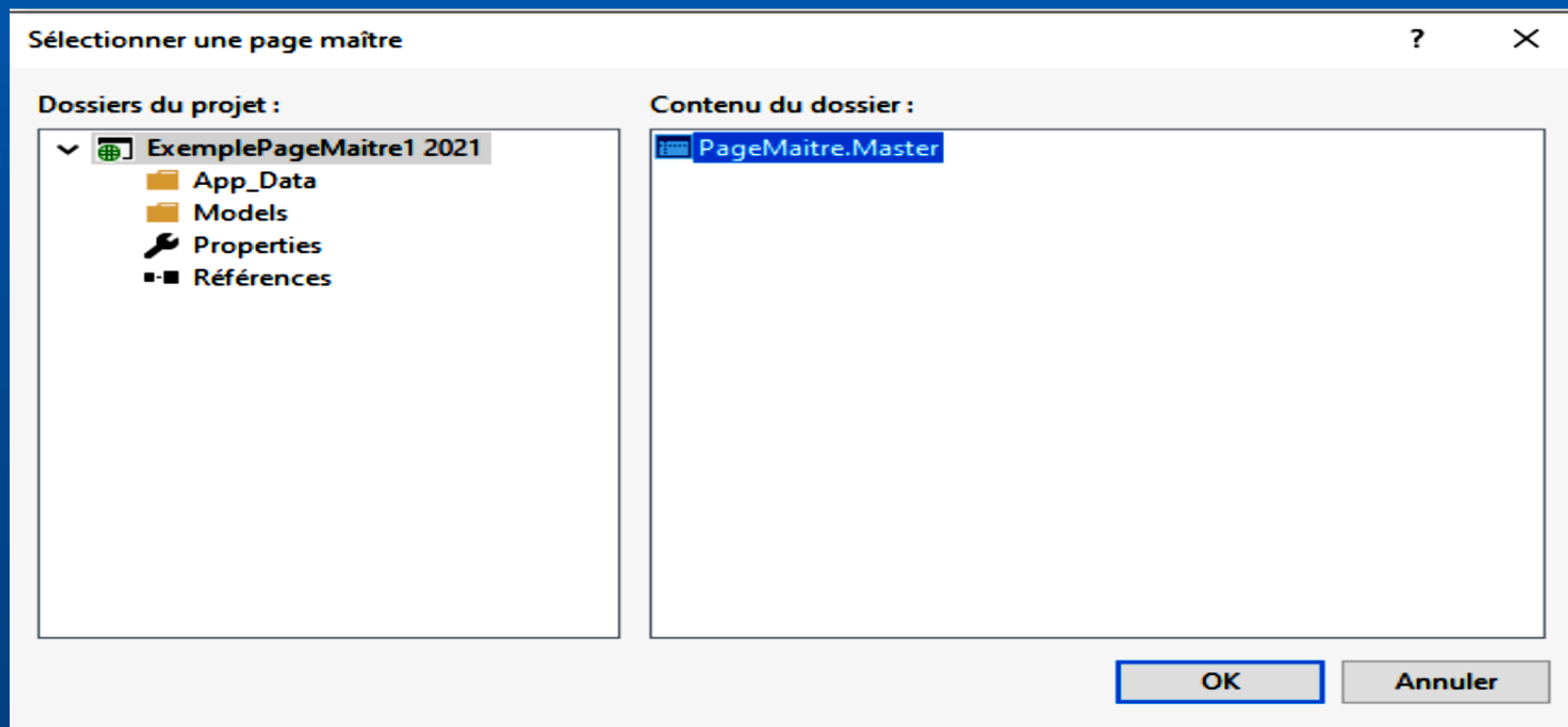
7. Créer une nouvelle page WebForm avec page maitre
8. cocher la case « sélectionner page maitre »
9. Choisir le page maitre déjà crée





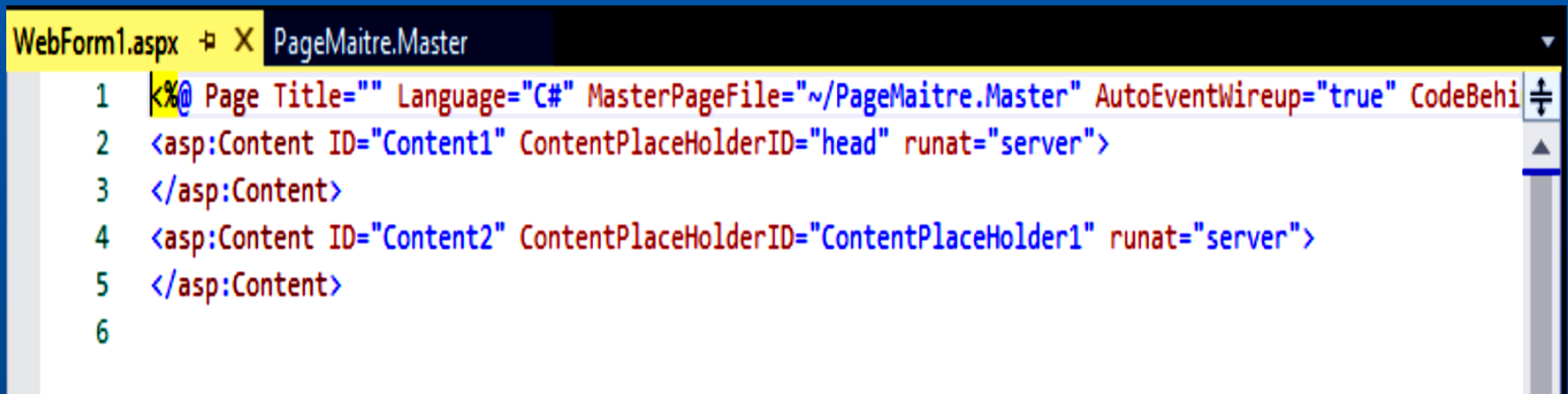
## Création de page maitre

7. cocher la case « sélectionner page maitre »
8. Choisir le page maitre déjà crée.

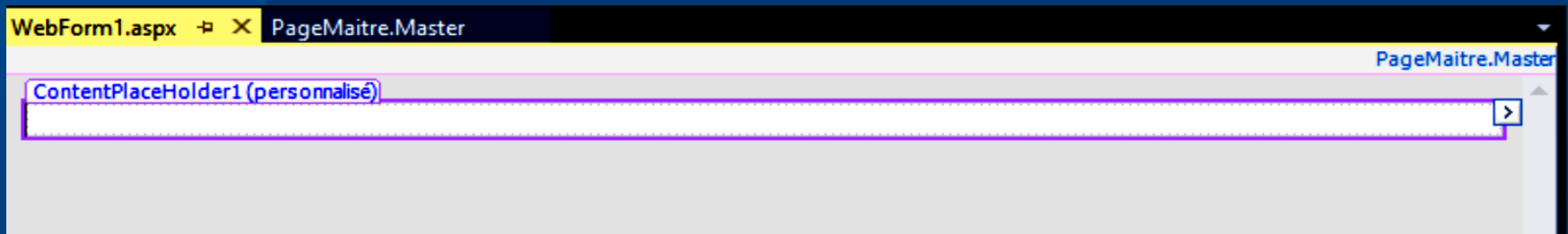


## Création de page maitre

*Vous pouvez alors remarquer que Visual Studio vous affiche une page qui est identique au master page de laquelle elle hérite, mais cette fois ci, tout le contenu du master page est "grisé" et non modifiable, et vous pouvez cependant modifier le contenu du, ou des ContentPlaceHolder intégré(s) dans votre page.*



```
WebForm1.aspx X PageMaitre.Master
1 <%@ Page Title="" Language="C#" MasterPageFile="~/PageMaitre.Master" AutoEventWireup="true" CodeBehi
2 <asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
3 </asp:Content>
4 <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
5 </asp:Content>
6
```



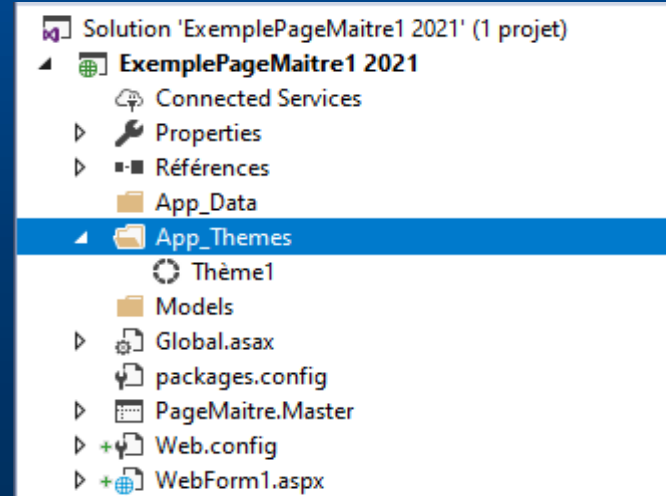
## *Les thèmes ASP.NET*

*Vous pouvez appliquer des thèmes à une page, à un site Web, ou au niveau global. La définition d'un thème au niveau du site Web applique des styles et des apparences (skins) à tous les contrôles et pages du site à moins que vous substituiez un thème pour une page individuelle. La définition d'un thème au niveau de la page applique des styles et des apparences à cette page ainsi qu'à tous ses contrôles. Par défaut, les thèmes se substituent aux paramètres des contrôles locaux. Vous pouvez également définir un thème comme thème de feuille de style, afin que le thème s'applique uniquement aux options du contrôle qui ne sont pas explicitement définies sur le contrôle.*

## Création d'un thème

Tous les thèmes en ASP.NET sont regroupés dans un même dossier: APP\_Themes. Par défaut il n'existe pas dans le projet. Pour l'ajouter faites un clic droit sur votre projet et choisissez Ajouter, puis faites "Ajouter le dossier ASP.NET" et enfin Theme.

Vous aurez dans votre explorateur de solution un dossier APP\_Themes qui devrait s'ajouter, contenant un autre dossier que Visual Studio va vous proposer de nommer. Pour ajouter un autre thème, faites un clic droit sur APP\_Themes, et ajouter un Nouveau dossier.

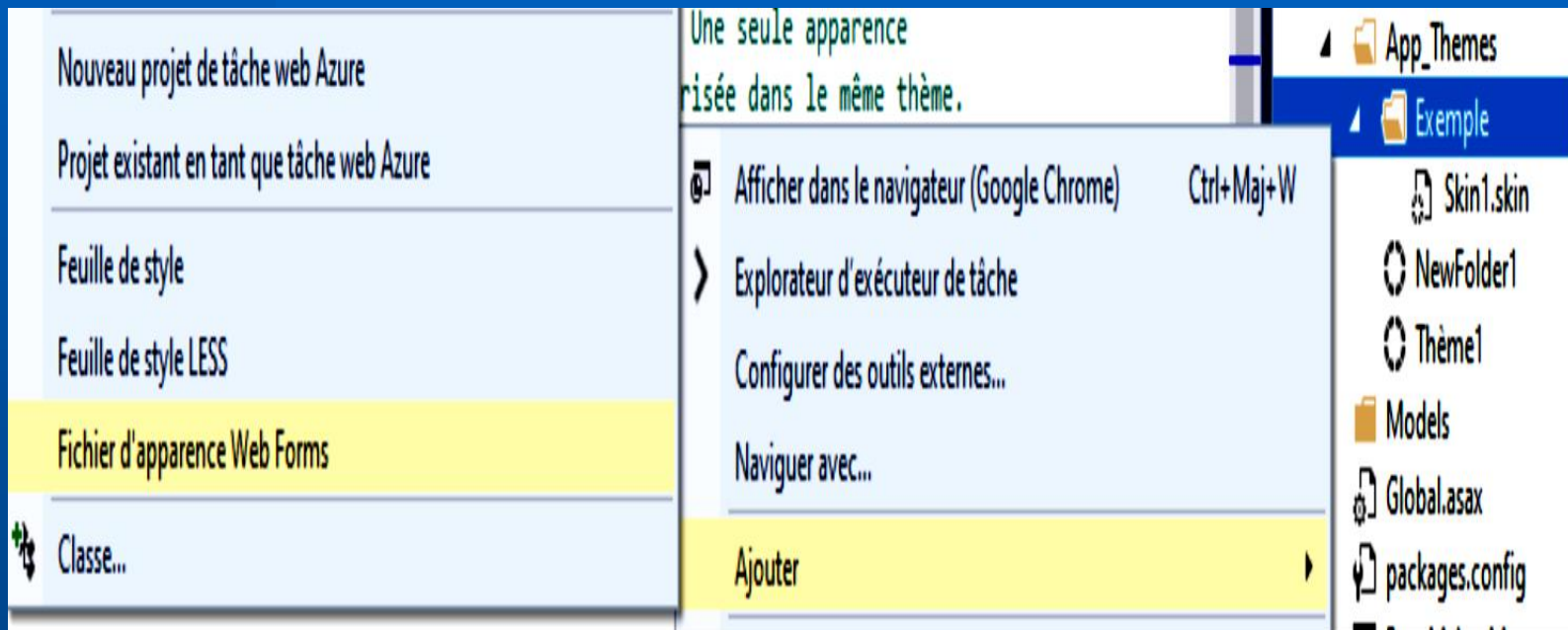


## Création d'un thème

*Pour ajouter des ressources dans un thème, il faut faire un clic droit dessus et choisir Ajouter puis Nouvel élément. On peut rajouter un fichier CSS, un skin, un fichier XML ... .*

*Le fichier css sert à définir le style pour du HTML ou des éléments ayant un ID ou une Class défini. Le fichier skin va, lui, permettre de définir le style pour un contrôle précisé. C'est-à-dire que l'on peut définir entièrement l'apparence des boutons (par exemple) et ainsi à chaque fois que l'on ajoutera un bouton qui possédera ce thème, le bouton aura déjà une apparence de défini. Ce système est très pratique puisque grâce à lui, une fois le style de nos contrôles défini, on ne s'occupe quasiment plus de l'apparence que chaque contrôle va prendre*

## Création fichier Skin



# Création fichier Skin

*Fichier skin ressemble par défaut lorsqu'on le créer :*

*Fichier skin : Skin1.skin*

```
<%--
```

Modèle d'apparence par défaut. Les apparences sont fournies uniquement à titre d'exemple.

1. Apparence de contrôle nommée. Le SkinId doit être défini de manière unique, car un type de contrôle d'un même thème ne peut pas compter de SkinId dupliqués.

```
<asp:GridView runat="server" SkinId="gridviewSkin" BackColor="White" >  
    <AlternatingRowStyle BackColor="Blue" />  
</asp:GridView>
```

2. Apparence par défaut. Le SkinId n'est pas défini. Une seule apparence de contrôle par défaut par type de contrôle est autorisée dans le même thème.

```
<asp:Image runat="server" ImageUrl="~/images/image1.jpg" />  
--%>
```

## Création fichier Skin

*Le fichier skin par défaut contient déjà des explications sur comment créer un skin. Il suffit donc de mettre le contrôle quasiment comme si c'était une page ASPX. Quasiment, parce que vous n'avez pas besoin de mettre les propriétés Text ect ... . La seule propriété obligatoire est le runat="server". Pour commencer ce skin nous allons y placer le skin du button que nous utiliserons dans la partie suivante :*

*Fichier skin : Skin1.skin*

```
<asp:Button runat="server" Font-Bold="true" ForeColor="Blue" />
```

*C'était un exemple de ce que l'on va avoir dans notre fichier skin. Il faut comprendre qu'on aura les skins de plusieurs contrôles sur chaque fichier skin (on ne fait pas un skin par contrôle).*

**Attention :** *Il ne faut pas qu'il y ait d'ID dans le skin.*



# Appliquer un thème

*Pour appliquer un thème à un site Web*

*Dans le Web.configon ajouter une propriété à la balise pages: theme comme dans l'exemple ci-dessous. "Exemple" correspond au nom du dossier contenant le thème dans APP\_Themes.*

*Web.config*

```
<system.web>  
  <pages theme="Exemple"/>  
</system.web>
```

*On peut aussi spécifier le thème dans la directive Page d'une page ASPX.*

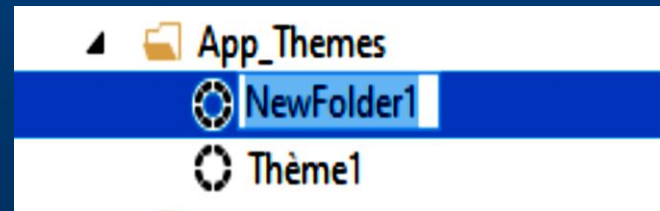
*Web.config*

```
<%@ Page Language="C#"  
  AutoEventWireup="true"  
  CodeBehind="Default.aspx.cs"  
  Theme="Exemple"  
  Inherits="WebApplication1.Default" %>
```

## *Appliquer un thème*

*Pour un groupe de pages on regroupe les pages qui doivent avoir un thème différent du reste du site dans un dossier. Clic droit sur le nom du projet, Ajouter, Nouveau dossier.*

*Dans ce dossier mettez les pages qui vous intéressent. Ensuite il va falloir rajouter un Web.config dans ce dossier. Ce Web.config ne s'appliquera que dans ce dossier. Pour cela ajouter un fichier de configuration web.*



## Exemple

*On utilise le thème et le skin créé plus tôt. Voici ce que contient notre page Default.aspx :*

*Web.config*

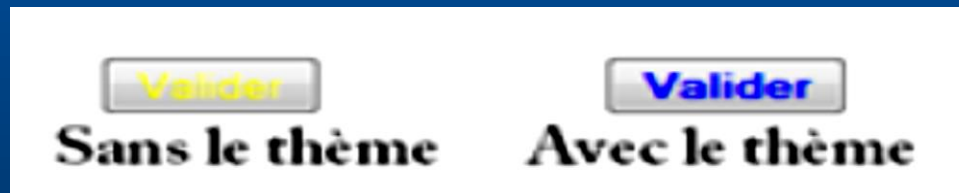
```
<%@ Page Language="C#"
    AutoEventWireup="true"
    CodeBehind="Default.aspx.cs"
    Theme="Exemple"
    Inherits="WebApplication1.Default" %>

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

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Page sans titre</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" Text="Valider" ForeColor="Yellow" />
        </div>
    </form>
</body>
</html>
```

## Exemple

*Vous remarquez que la couleur du texte du bouton est bleu comme défini dans le thème malgré la spécification du jaune dans la balise. Maintenant enlevez la propriété Theme dans la directive Page. Vous remarquerez alors que le texte du bouton est maintenant jaune (voir l'image ci-contre). Créer une page Page1.aspx contenant le formulaire suivant :*



## *Exercice*

*Créer un mini-site respectant le design ci-après et ayant la structure suivante :*

- ✓ *Une MasterPage contenant le Header, le Menu et le Footer*
- ✓ *3 pages de site (Accueil, Profil, Contact)*
- ✓ *Un thème pour chaque saison avec couleurs et images background correspondants à chaque saison de l'année.*
- ✓ *Une liste déroutante pour basculer dynamiquement vers le thème d'une saison*

# Exercice



## Introduction aux saisons

Une année est composée de 12 mois, qui se divisent en quatre saisons. Les saisons dépendent de la position de la Terre par rapport au Soleil selon les mois de l'année. Il existe quatre saisons :

- L'automne
- L'hiver
- Le printemps
- L'été

Selon les saisons, le Soleil est plus ou moins haut dans le ciel et les quantités de chaleur et de pluies reçues sont différentes. C'est pourquoi le rythme des saisons transforme la nature, car les plantes dépendent directement du temps qu'il fait.