Programmation web avec ASP.Net

Réalisée par : Karima BELABED

Contenu

Introduction

ASP.NET est un Framework d'applications Web développé et commercialisé par Microsoft pour permettre aux programmeurs de créer des sites Web dynamiques. Il vous permet d'utiliser un langage de programmation complet tel que C # ou VB.NET pour créer facilement des applications Web.

Les codes d'application ASP.NET peuvent être écrits dans l'une des langues suivantes:

- C #
- Visual Basic.Net
- Jscript
- J #

ASP.NET est utilisé pour produire des applications Web interactives, pilotées par les données, sur Internet. Il se compose d'un grand nombre de contrôles tels que des zones de texte, des boutons et des étiquettes pour assembler, configurer et manipuler du code pour créer des pages HTML.

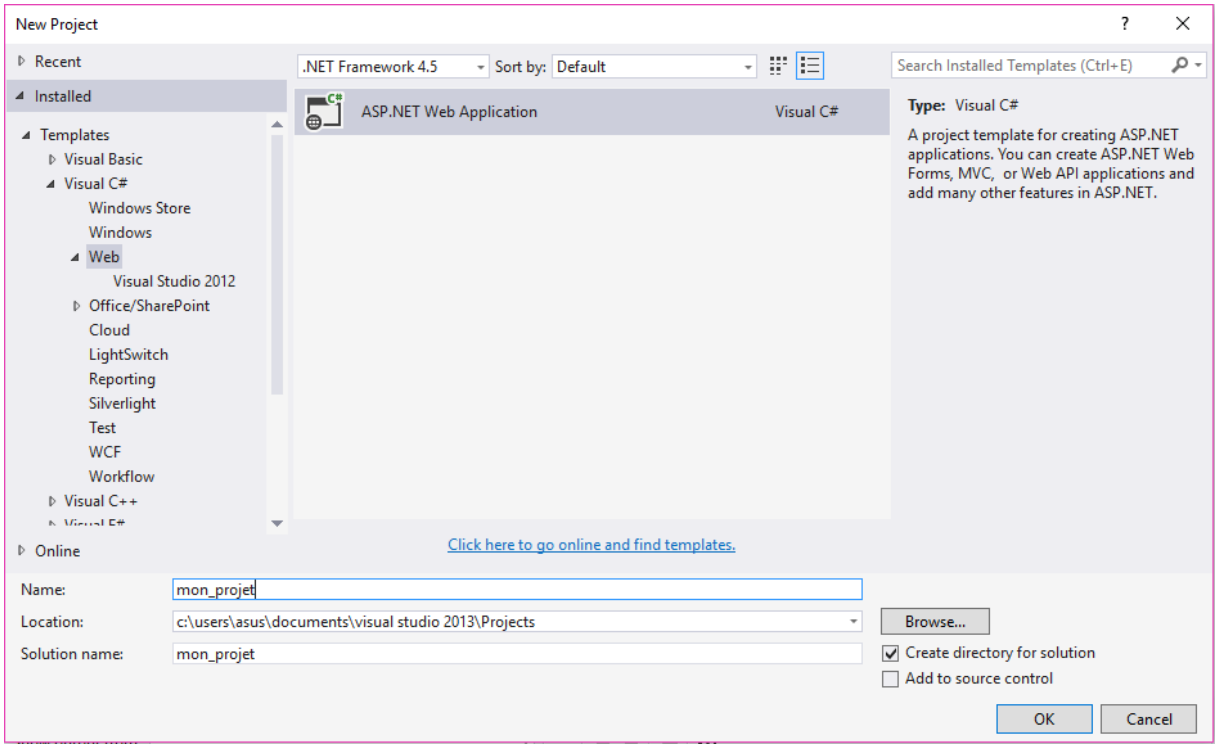
Création d'un projet ASP.NET

L'outil de développement clé pour la création d'applications et de frontaux ASP.NET est Visual Studio. Dans ce tutoriel, nous travaillons avec Visual Studio 2013.

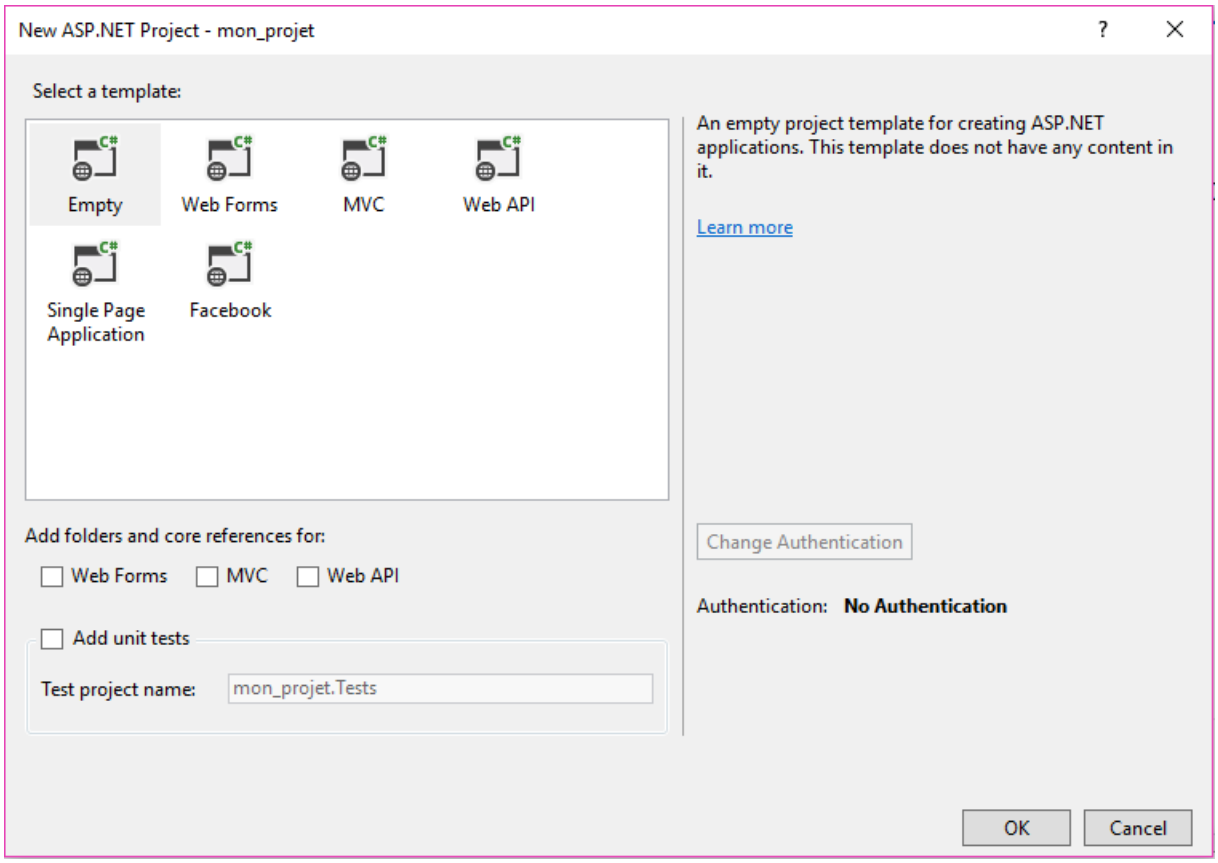
Visual Studio est un environnement de développement intégré pour l'écriture, la compilation et le débogage du code. Il fournit un ensemble complet d'outils de développement pour la création d'applications Web ASP.NET, de services Web, d'applications de bureau et d'applications mobiles.

Étape 1 – Ouvrez le menu Fichier et sélectionnez Nouveau > Projet.

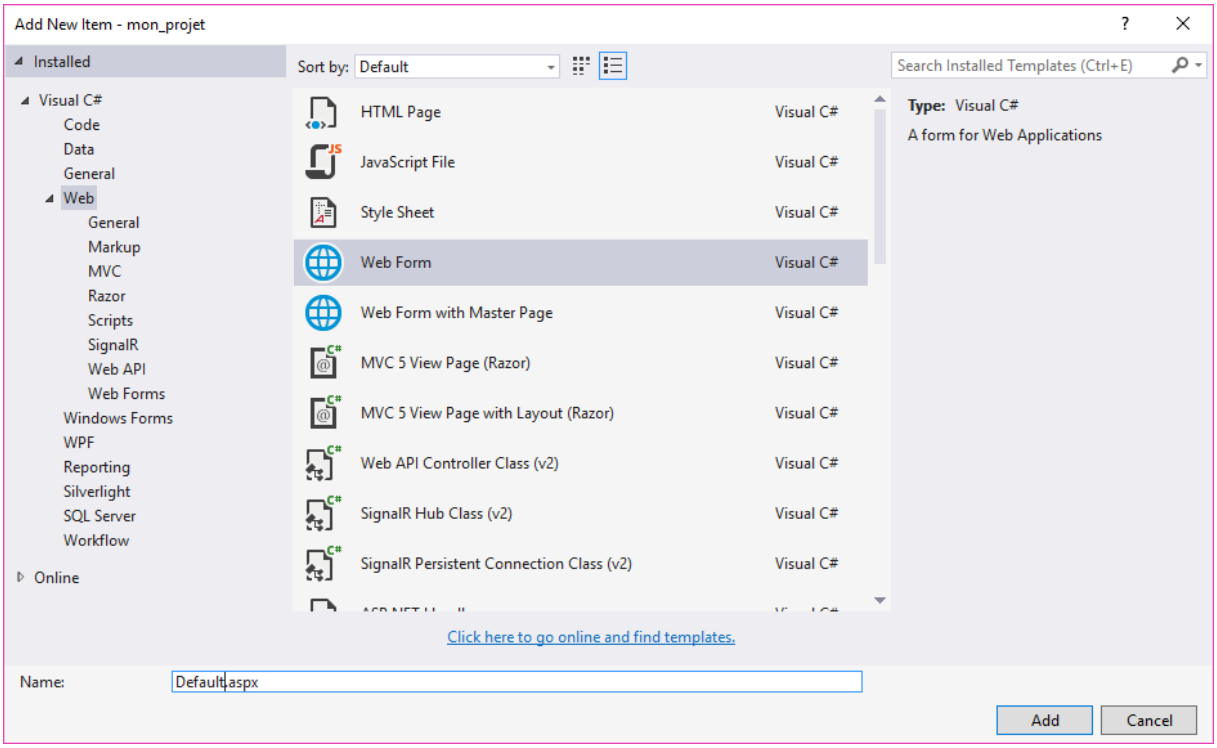
Étape 2 – Choisissez un modèle d'application ; sélectionnez C# dans la liste déroulante, Web > ASP.NET Web Application et Nommez votre projet.



Étape 3 – Sélectionnez Empty, cliquez sur le bouton OK pour créer ce nouveau projet.



Étape 4 – Cliquez avec le bouton droit sur le projet Ajouter > Web Form. Vous verrez s'afficher la boîte de dialogue suivante :



Lorsque vous démarrez un nouveau site Web, ASP.NET fournit les dossiers de démarrage et les fichiers du site, y compris deux fichiers pour le premier formulaire Web du site.

Le fichier nommé Default.aspx contient le code HTML et ASP qui définit le formulaire :

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

<!DOCTYPE html>

<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>
```

Le fichier nommé Default.aspx.cs contient le code dans la langue que vous avez choisi et ce code est responsable des actions effectuées sur un formulaire.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace mon_projet
{
    1 reference
    public partial class Default : System.Web.UI.Page
    {
        0 references
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Behind

Le code behind est le code dans lequel on écrira les instructions qui seront exécutées sur le serveur comme, par exemple, récupérer une valeur, changer un attribut dynamiquement, créer un objet ….

Les en-têtes / directives

L'ASP.NET a une ligne qui définit certains paramètres et qui ressemble par défaut à ce qui suit (en C#) :

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

On appelle ceci une directive (plus précisément, il s'agit ici de la directive Page).
Il y est indiqué que nous allons utiliser le C# comme langage (Language="C#"), que le nom de la page du code behind, correspondant à cette page .aspx, s'appelle Default.aspx.cs (CodeBehind="Default.aspx.cs"). Est aussi définit le namespace et le nom de la classe lié à la page (Inherits="NouveauProjet._Default").

Projets et solutions

Un projet ASP.NET contient les fichiers de contenu suivants :

- Fichier de page (.aspx)
- Contrôle utilisateur (.ascx)
- Service Web (.asmx)
- Page maître (.master)
- Plan du site (.sitemap)
- Fichier de configuration du site Web (.config)

Construire et gérer un projet

Vous pouvez exécuter une application en:

- Sélection du début
- En sélectionnant Démarrer sans déboguer dans le menu Déboguer,
- appuyant sur F5
- Ctrl-F5

Le programme a un sens construit, les fichiers .exe ou .dll sont générés en sélectionnant une commande dans le menu Générer.

Le cycle de vie ASP.NET spécifie comment:

- ASP.NET traite les pages pour produire une sortie dynamique
- L'application et ses pages sont instanciés et traitées
- ASP.NET compile les pages dynamiquement

Le cycle de vie ASP.NET peut être divisé en deux groupes:

- Cycle de vie de l'application :
  - L'utilisateur fait une demande d'accès à la ressource de l'application, une page. Le navigateur envoie cette requête au serveur Web.
  - Un pipeline unifié reçoit la première requête et les événements suivants ont lieu:
    - ✓ Un objet de la classe ApplicationManager est créé.
    - ✓ Un objet de la classe HostingEnvironment est créé pour fournir des informations concernant les ressources.
    - ✓ Les éléments de premier niveau de l'application sont compilés.
  - Les objets de réponse sont créés. Les objets d'application tels que HttpContext, HttpRequest et HttpResponse sont créés et initialisés.
  - Une instance de l'objet HttpApplication est créée et affectée à la demande.
  - La requête est traitée par la classe HttpApplication. Différents événements sont soulevés par cette classe pour le traitement de la requête.

- Cycle de vie de la page :

  Lorsqu'une page est demandée, elle est chargée dans la mémoire du serveur, traitée et envoyée au navigateur. Ensuite, il est déchargé de la mémoire. À chacune de ces étapes, des méthodes et des événements sont disponibles, lesquels peuvent être remplacés en fonction des besoins de l'application. En d'autres termes, vous pouvez écrire votre propre code pour remplacer le code par défaut.

Événements de cycle de vie d'une page ASP.NET

À chaque étape du cycle de vie de la page, la page soulève des événements qui pourraient être codés. Un gestionnaire d'événements est essentiellement une fonction ou un sous-programme, lié à l'événement, utilisant des attributs déclaratifs tels qu'Onclick ou handle.
Voici la séquence des événements qui sont déclenchés chaque fois que vous demandez une page:
PreInit

Init
InitComplete
Pré-chargement
Charge
LoadComplete
PreRender
PreRenderComplete
SaveStateComplete
Décharger

Propriétés de la page

La classe Page a un certain nombre de propriétés importantes. Certains des plus utiles sont énumérés ci-dessous.

EnableViewState indique si la page conserve l'état d'affichage pour elle-même et ses contrôles. Vous pouvez obtenir ou définir cette propriété. La valeur par défaut est true, l'état d'affichage est conservé.

ErrorPage spécifie la page d'erreur à laquelle le navigateur doit être redirigé au cas où une exception non gérée se produirait.

IsPostBack indique si la page est en cours de chargement en réponse à une publication du client ou est en cours de chargement pour la première fois.

```
protected void Page_Load(object sender, EventArgs e)
{
    /*juste lors du premier chargement*/
    if (Page.IsPostBack)//demander de recharger la page
    {
        label1.Text += "page rechargée <br/>";
    }
}
```

page rechargée
page rechargée
page rechargée
page rechargée
page rechargée

cliquer

IsValid indique si la validation de la page a réussi.

Request obtient l'objet HTTP Request, qui vous permet d'accéder aux données des demandes HTTP entrantes.

La réponse obtient l'objet de réponse HTTP, qui vous permet d'envoyer des données de réponse à un navigateur.

Session obtient l'objet Session en cours, fourni par ASP.NET pour stocker l'état de la session.

Trace obtient un objet TraceContext pour la page, que vous pouvez utiliser pour écrire des informations de trace.

Vue d'ensemble des contrôles ASP.NET :

Contrôles standard : vous permettent de rendre des éléments de formulaire standard tels que des boutons, des champs de saisie et des étiquettes…

La balise ASP pour un contrôle de **bouton**:

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

La balise ASP pour un contrôle de **texte**:

```
<asp:TextBox ID="txtMsg" runat="server" ></asp:TextBox>
```

La balise ASP pour un contrôle de la **case à cocher**:

```
<asp:CheckBox ID= "checkS" runat= "Server"></asp:CheckBox>
```

La balise ASP pour un contrôle de **bouton radio**:

```
<asp:RadioButton ID= "radio1" runat= "Server"></asp: RadioButton>
```

La balise ASP pour un contrôle de **zone de liste**:

```
<asp:ListBox ID="ListBox1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged"></asp:ListBox>
```

La balise ASP pour un contrôle de **la liste déroulante**:

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged"></asp:DropDownList>
```

La balise ASP pour un contrôle de **la liste des boutons radio**:

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True"
  OnSelectedIndexChanged="RadioButtonList1_SelectedIndexChanged">
</asp:RadioButtonList>
```

La balise ASP pour un contrôle de **la liste de cases à cocher**:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server" AutoPostBack="True"
  OnSelectedIndexChanged="CheckBoxList1_SelectedIndexChanged">
</asp:CheckBoxList>
```

La balise ASP pour un contrôle d'**une liste à puces**:

```
<asp:BulletedList ID="BulletedList1" runat="server"></asp:BulletedList>
```

La balise ASP pour un contrôle de **lien hypertexte**:

```
<asp:HyperLink ID="HyperLink1" runat="server"> HyperLink</asp:HyperLink>
```

La balise ASP pour un contrôle d'**image**:

```
<asp:Image ID="Image1" runat="server">
```

Contrôles de validation : vous permettent de valider les données de formulaire avant de soumettre les données au serveur.

Pour utiliser Contrôles de validation avec ASP.NET, une ligne doit être présente dans le fichier de configuration [Web.config] :

```
<appSettings>
   <add key="ValidationSettings:UnobtrusiveValidationMode" value="None" />
</appSettings>
```

RequiredFieldValidator - Vous permet d'obliger un utilisateur à entrer une valeur dans un champ de formulaire.

```
<asp:RequiredFieldValidator ID="rfvcandidate"
  runat="server" ControlToValidate ="ddlcandidate"
  ErrorMessage="Please choose a candidate"
  InitialValue="Please choose a candidate">
</asp:RequiredFieldValidator>
```

- **ControlToValidate :** L'ID du champ de formulaire en cours de validation.
- **ErrorMessage :** Le message d'erreur s'affiche lorsque la validation échoue.

RangeValidator - Vous permet de vérifier si une valeur se situe entre une valeur minimum et une valeur maximum.

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtage"
  ErrorMessage="Enter votre age (18 - 40)" MaximumValue="40"
  MinimumValue="18" Type="Integer">
  </asp:RangeValidator>
```

- **ControlToValidate :** L'ID du champ de formulaire en cours de validation.
- **ErrorMessage :** Le message d'erreur s'affiche lorsque la validation échoue.
- **MinimumValue :** La valeur minimale de la plage de validation.
- **MaximumValue :** Valeur maximale de la plage de validation.

- **Type :** Le type de comparaison à effectuer. Les valeurs possibles sont String, Integer, Double, Date et Currency.

CompareValidator - Permet de comparer une valeur avec une autre valeur ou d'effectuer une vérification de type de données.

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
  ErrorMessage="La date de fin doit être supérieure à la date de début"
  ControlToValidate="txtDateFin" ControlToCompare="txtDateDebut" Type="Date"
   Operator=" GreaterThan">
</asp:CompareValidator>
```

- **ControlToValidate :** L'ID du champ de formulaire en cours de validation.
- **ErrorMessage :** Le message d'erreur s'affiche lorsque la validation échoue.
- **Type :** Le type de valeur comparé. Les valeurs possibles sont String, Integer, Double, Date et Currency.
- **Operator :** Le type de comparaison à effectuer.
- **ValueToCompare :** La valeur fixe par rapport à laquelle comparer.
- **ControlToCompare :** L'ID d'un contrôle par rapport auquel comparer.

RegularExpressionValidator - Vous permet de comparer une valeur avec une expression régulière.

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
  ValidationExpression="string" ValidationGroup="string">
</asp:RegularExpressionValidator>
```

- **ValidationExpression :** L'expression régulière.

CustomValidator - Vous permet d'effectuer une validation personnalisée.

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
  ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">
</asp:CustomValidator>
```

- **ClientValidationFunction :** La fonction de validation JavaScript définie dans la balise <head> de la page.

ValidationSummary - Vous permet d'afficher un résumé de toutes les erreurs de validation dans une page.

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
  DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

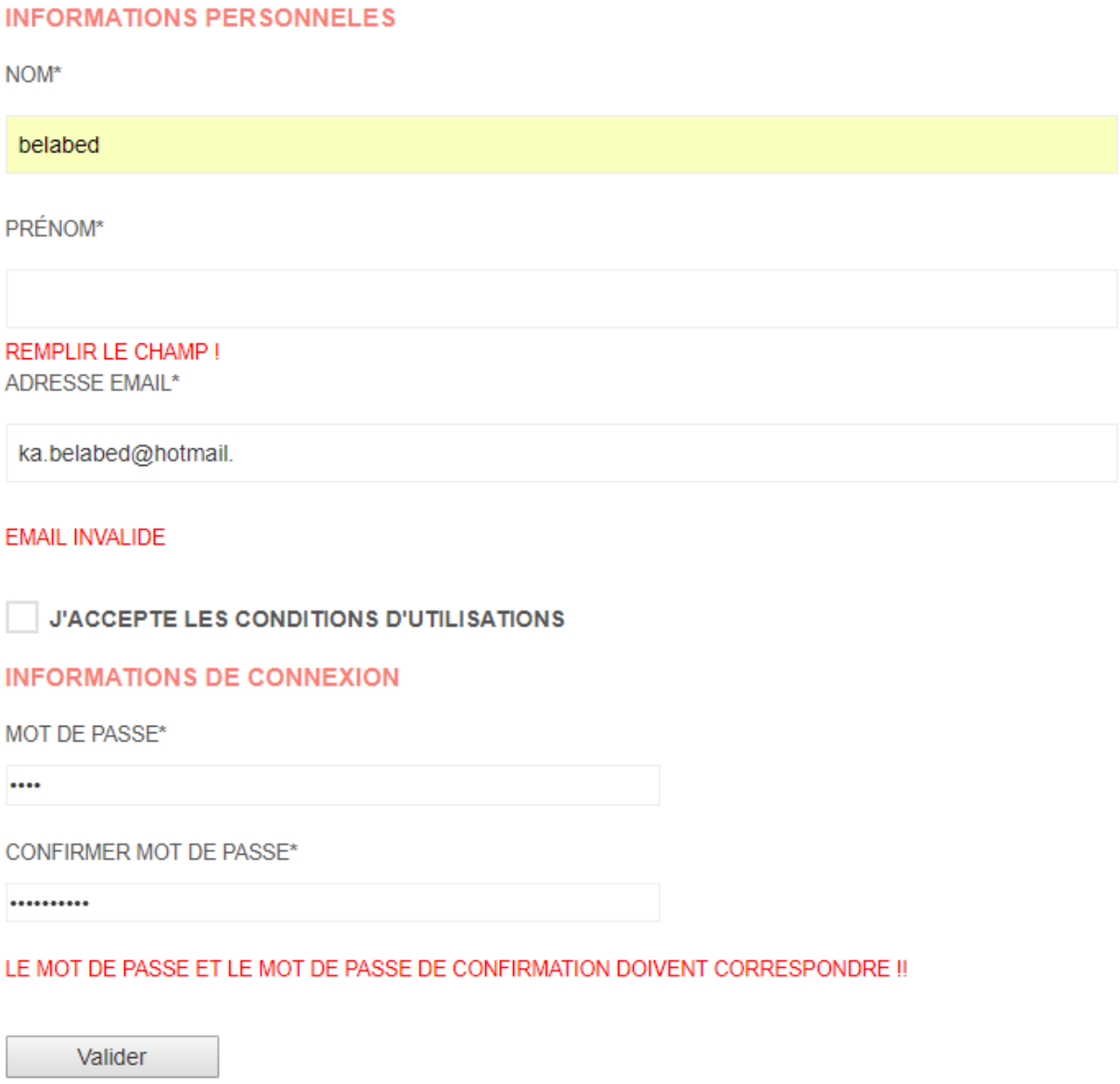
- **DisplayMode :** Vous permet de spécifier comment les messages d'erreur sont formatés. Les valeurs possibles sont BulletList , List et SingleParagraph .
- **HeaderText :** Vous permet d'afficher le texte de l'en-tête au-dessus du résumé de validation.
- **ShowMessageBox :** Vous permet d'afficher une boîte d'alerte contextuelle.
- **ShowSummary :** Vous permet de masquer le résumé de validation dans la page.

Exemple de contrôles de validation :

```
<h3>INFORMATIONS PERSONNELES</h3>
    <span>Nom<label>*</label></span>
    <asp:TextBox ID="txtnom" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="req1" runat="server" ControlToValidate="txtnom" ErrorMessage="Remplir le champ !" ForeColor="red">
    </asp:RequiredFieldValidator>
    <span>Prénom<label>*</label></span>
    <asp:TextBox ID="txtprenom" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="req2" runat="server" ControlToValidate="txtprenom" ErrorMessage="Remplir le champ !" ForeColor="red">
    </asp:RequiredFieldValidator>
     <span>Adresse Email<label>*</label></span>
     <asp:TextBox ID="txtemail" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="req3" runat="server" ControlToValidate="txtemail" ErrorMessage="Remplir le champ !" ForeColor="red">
    </asp:RequiredFieldValidator>
    <asp:RegularExpressionValidator ID="reg1" runat="server" ControlToValidate="txtemail"
        ValidationExpression="^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]{2,}[.][a-zA-Z]{2,4}$" ErrorMessage="Email invalide" ForeColor="Red">
    </asp:RegularExpressionValidator>
     <label class="checkbox"><asp:CheckBox ID="CheckBox1" runat="server" /><i> </i>J'accepte les conditions d'utilisations</label>
        <h3>INFORMATIONS DE CONNEXION</h3>
            <span>Mot de passe<label>*</label></span>
            <asp:TextBox ID="txtmdp" runat="server" TextMode="Password"></asp:TextBox>
    <asp:RequiredFieldValidator ID="req4" runat="server" ControlToValidate="txtmdp" ErrorMessage="Remplir le champ !" ForeColor="red">
    </asp:RequiredFieldValidator>
            <span>Confirmer mot de passe<label>*</label></span>
            <asp:TextBox ID="txtcmdp" runat="server" TextMode="Password"></asp:TextBox>
    <asp:RequiredFieldValidator ID="req5" runat="server" ControlToValidate="txtcmdp" ErrorMessage="Remplir le champ !" ForeColor="red">
    </asp:RequiredFieldValidator>
            <asp:CompareValidator ID="cmp1" runat="server" ControlToValidate="txtmdp" ControlToCompare="txtcmdp" Operator="Equal"
                ErrorMessage="le mot de passe et le mot de passe de confirmation doivent correspondre !!" ForeColor="Red">
            </asp:CompareValidator>
   <asp:Button ID="btn_valider" runat="server" Text="Valider" OnClick="btn_valider_Click"  />
```

Un contrôle RequiredFieldValidator distinct est associé à chacun des cinq champs de formulaire.
Si vous tentez de soumettre le formulaire sans entrer de valeur pour un champ, un message d'erreur de validation s'affiche.
Un contrôle CompareValidator est utilisé pour vérifier si les champs de texte txtmdp et le txtcmdp contient une même valeur.

Un contrôle RegularExpressionValidator est utilisé pour comparer la valeur de champ d'adresse email avec l'expression régulière.
Vous pouvez utiliser une expression régulière pour représenter les modèles de chaîne tels que les adresses électroniques, les numéros de sécurité sociale, les numéros de téléphone, les dates, les montants monétaires et les codes de produit.



Contrôles enrichis : vous permettent d'afficher des éléments tels que des calendriers, des boutons de téléchargement de fichiers, des bannières publicitaires rotatives et des assistants multi-étapes.

La balise ASP pour un contrôle de **Calendrier** :

```
<asp:Calender ID = "Calendar1" runat = "server">
</asp:Calender>
```

La balise ASP pour un contrôle d'**AdRotator** : sélectionne aléatoirement des graphiques de bannière à partir d'une liste, qui est spécifiée dans un fichier de planification XML externe.

```
<asp:AdRotator  runat = "server" AdvertisementFile = "adfile.xml"
Target = "_blank" />
```

La balise ASP pour un contrôle de **FileUpload** : Permet à l'utilisateur de rechercher et de sélectionner le fichier à télécharger, en fournissant un bouton de navigation et une zone de texte pour entrer le nom de fichier.

```
<asp:FileUpload ID= "Uploader" runat = "server" />
```

Contrôles de données : vous permettent de travailler avec des données telles que des données de base de données.

- ✓ **Un contrôle de source de données -** Il gère la connexion aux données, la sélection des données et d'autres tâches telles que la pagination et la mise en cache des données, etc.
- ✓ La balise ASP pour un contrôle de **SqlDataSource** : La connexion aux données s'effectue via deux propriétés importantes ConnectionString et ProviderName.
  <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:gestion %>" SelectCommand="SELECT * FROM [NomTable]"></asp:SqlDataSource>
- ✓ **Un contrôle de vue de données -** Il lie et affiche les données et permet la manipulation des données.

Commandes de navigation : vous permettent d'afficher les éléments de navigation standard telle que les menus, les vues arborescentes.

La balise ASP pour un contrôle de **MultiView** et **View** : vous permettent de diviser le contenu d'une page en différents groupes, affichant un seul groupe à la fois.

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

La balise ASP pour un contrôle de **TreeView** : permet d'afficher des données hiérarchiques dans une structure arborescente.

```
<asp: TreeView ID= "TreeView1" runat= "server">
  <Nodes>   <asp: treeNode Text= "View1" ToolTip= "titre">
     <asp: treeNode Text= "View1" NavigateUrl= "page1.aspx" Target= "_blank">
     <asp: treeNode Text= "View1" NavigateUrl= "page2.aspx" Target= "_blank">
     </asp: treeNode>
  </Nodes>
</asp:MultiView>
```

La balise ASP pour un contrôle de **Menu** :

```
<asp: Menu ID= "TreeView1" runat= "server">
  <Article>
     <asp: MenuItem Text= "fichier" Value= "fichier">
       <asp: MenuItem Text= "ouvrir" Value= "ouvrir"></asp: MenuItem >
       <asp: MenuItem Text= "nouveau" Value= "nouveau"></asp: MenuItem >
     </asp: MenuItem >
  </Article>
</asp: Menu >
```

Commandes de connexion : vous permettent d'afficher le nom d'utilisateur, le mot de passe et les formulaires d'inscription.

Contrôles HTML : vous permettent de convertir n'importe quelle balise HTML en contrôle côté serveur. (voir le cours de HTML).

Propriétés des contrôles serveur

Propriété	La description
AccessKey	Appuyez sur cette touche avec la touche Alt pour déplacer le focus sur le contrôle.
BackColor	Couleur de fond.
BindingContainer	Le contrôle qui contient la liaison de données de ce contrôle.
BorderColor	Couleur de la bordure.
BorderStyle	Style de bordure.
BorderWidth	Largeur de la bordure.
ClientID	ID de contrôle pour le balisage HTML.

Controls	Collection de tous les contrôles contenus dans le contrôle.
CssClass	Classe CSS
DesignMode	Indique si le contrôle est utilisé sur une surface de conception.
DisabledCssClass	Obtient ou définit la classe CSS à appliquer à l'élément HTML rendu lorsque le contrôle est désactivé.
Enabled	Indique si le contrôle est grisé.
EnableViewState	Indique si l'état d'affichage du contrôle est maintenu.
Events	Obtient une liste de délégués du gestionnaire d'événements pour le contrôle.
Font	Police de caractère.
ForeColor	Couleur de premier plan.
Height	Hauteur en pixels ou en%
ID	Identifiant pour le contrôle.
IsChildControlStateCleared	Indique si les contrôles contenus dans ce contrôle ont un état de contrôle.
IsEnabled	Obtient une valeur indiquant si le contrôle est activé.
Page	Page contenant le contrôle.
Parent	Contrôle parental.
Site	Le conteneur qui héberge le contrôle en cours lorsqu'il est rendu sur une surface de conception.

SkinID	Obtient ou définit l'habillage à appliquer au contrôle.
Style	Obtient une collection d'attributs de texte qui seront rendus en tant qu'attribut de style sur la balise externe du contrôle serveur Web.
TabIndex	Obtient ou définit l'index de tabulation du contrôle serveur Web.
ViewState	Obtient un dictionnaire d'informations d'état qui enregistre et restaure l'état d'affichage d'un contrôle serveur sur plusieurs demandes pour la même page.
Visible	Indique si un contrôle serveur est visible.
Width	Obtient ou définit la largeur du contrôle du serveur Web.

Méthodes des contrôles serveur :

Méthode	La description
AddAttributesToRender	Ajoute les attributs HTML et les styles qui doivent être rendus au HtmlTextWriterTag spécifié.
AddedControl	Appelée après l'ajout d'un contrôle enfant à la collection Controls de l'objet de contrôle.
AddParsedSubObject	Indique au contrôle serveur qu'un élément, XML ou HTML, a été analysé et ajoute l'élément à la collection de contrôles du contrôle serveur.
ApplyStyleSheetSkin	Applique les propriétés de style définies dans la feuille de style de page au contrôle.
ClearCachedClientID	Infrastructure. Définit la valeur ClientID en cache sur null.
ClearChildControlState	Supprime les informations sur l'état du contrôle pour les contrôles enfants du contrôle serveur.

ClearChildState	Supprime les informations sur l'état d'affichage et l'état du contrôle pour tous les contrôles enfants du contrôle serveur.
ClearChildViewState	Supprime les informations d'état d'affichage pour tous les contrôles enfants du contrôle serveur.
CreateChildControls	Utilisé dans la création de contrôles enfants.
CreateControlCollection	Crée un nouvel objet ControlCollection pour contenir les contrôles enfants.
CreateControlStyle	Crée l'objet de style utilisé pour implémenter toutes les propriétés liées au style.
DataBind	Lie une source de données au contrôle serveur et à tous ses contrôles enfants.
DataBind (Booléen)	Lie une source de données au contrôle serveur et à tous ses contrôles enfants avec une option permettant de déclencher l'événement DataBinding.
DataBindChildren	Lie une source de données aux contrôles enfants du contrôle serveur.
Disposer	Permet à un contrôle serveur d'effectuer un nettoyage final avant qu'il ne soit libéré de la mémoire.
EnsureChildControls	Détermine si le contrôle serveur contient des contrôles enfants. Si ce n'est pas le cas, il crée des contrôles enfants.
EnsureID	Crée un identifiant pour les contrôles qui n'ont pas d'identifiant.
Est égal à (objet)	Détermine si l'objet spécifié est égal à l'objet actuel.
Finaliser	Autorise un objet à tenter de libérer des ressources et d'effectuer d'autres opérations de nettoyage avant que l'objet ne soit récupéré par la récupération de place.

FindControl (Chaîne)	Recherche dans le conteneur de dénomination actuel un contrôle serveur avec le paramètre id spécifié.
FindControl (String, Int32)	Recherche dans le conteneur de dénomination actuel un contrôle serveur avec l'ID spécifié et un entier.
Concentrer	Définit le focus d'entrée sur un contrôle.
GetDesignModeState	Obtient des données de conception pour un contrôle.
GetType	Obtient le type de l'instance en cours.
GetUniqueIDRelativeTo	Renvoie la partie préfixée de la propriété UniqueID du contrôle spécifié.
HasControls	Détermine si le contrôle serveur contient des contrôles enfants.
HasEvents	Indique si les événements sont enregistrés pour le contrôle ou pour les contrôles enfants.
IsLiteralContent	Détermine si le contrôle serveur contient uniquement du contenu littéral.
LoadControlState	Restaure les informations de l'état du contrôle.
LoadViewState	Restaure les informations d'état d'affichage.
MapPathSecure	Récupère le chemin physique auquel un chemin virtuel, absolu ou relatif, correspond.
MemberwiseClone	Crée une copie superficielle de l'objet actuel.
MergeStyle	Copie tous les éléments non vides du style spécifié dans le contrôle Web, mais n'écrase aucun élément de style existant du contrôle.
OnBubbleEvent	Détermine si l'événement du contrôle serveur est transmis à la hiérarchie de contrôle du serveur d'interface utilisateur de la page.

OnDataBinding	Déclenche l'événement de liaison de données.
OnInit	Déclenche l'événement Init.
En charge	Déclenche l'événement Load.
OnPreRender	Déclenche l'événement PreRender.
OnUnload	Déclenche l'événement Unload.
Fichier ouvert	Obtient un flux utilisé pour lire un fichier.
SupprimeContrôle	Appelé après qu'un contrôle enfant a été supprimé de la collection de contrôles de l'objet de contrôle.
Rendre	Rend le contrôle à l'auteur HTML spécifié.
RenderBeginTag	Restitue la balise d'ouverture HTML du contrôle au writer spécifié.
RenderChildren	Envoie le contenu des enfants d'un contrôle serveur à un objet HtmlTextWriter fourni, qui écrit le contenu à rendre sur le client.
RenderContents	Rend le contenu du contrôle à l'auteur spécifié.
RenderControl (HtmlTextWriter)	Exporte le contenu du contrôle serveur vers un objet HtmlTextWriter fourni et stocke les informations de suivi concernant le contrôle si le suivi est activé.
RenderEndTag	Restitue la balise de fermeture HTML du contrôle dans le writer spécifié.
ResolveAdapter	Obtient l'adaptateur de contrôle responsable du rendu du contrôle spécifié.
SaveControlState	Enregistre les changements d'état de contrôle du serveur qui se sont produits depuis le moment où la page a été publiée sur le serveur.

SaveViewState	Enregistre tout état qui a été modifié après l'appel de la méthode TrackViewState.
SetDesignModeState	Définit les données de conception pour un contrôle.
ToString	Retourne une chaîne qui représente l'objet actuel.
TrackViewState	Force le contrôle à suivre les modifications apportées à son état d'affichage afin qu'elles puissent être stockées dans la propriété d'état d'affichage de l'objet.

ASP.net et SQL Server:

ADO.NET

C'est la technologie utilisée pour travailler avec des données et des bases de données. Il permet d'accéder à des sources de données telles que le serveur SQL, OLE DB, XML, etc. ADO.NET permet la connexion aux sources de données pour la récupération, la manipulation et la mise à jour des données.

DataSet

C'est un objet qui réside en mémoire et qui correspond à une copie locale des données d'une classe.il contient (DataTable) les tables d'une base, mais aussi les relations entre les tables et les contraintes appliquées aux données(DataRelation).

L'objet **DataRow** représente une ligne dans une table.

DataAdapter

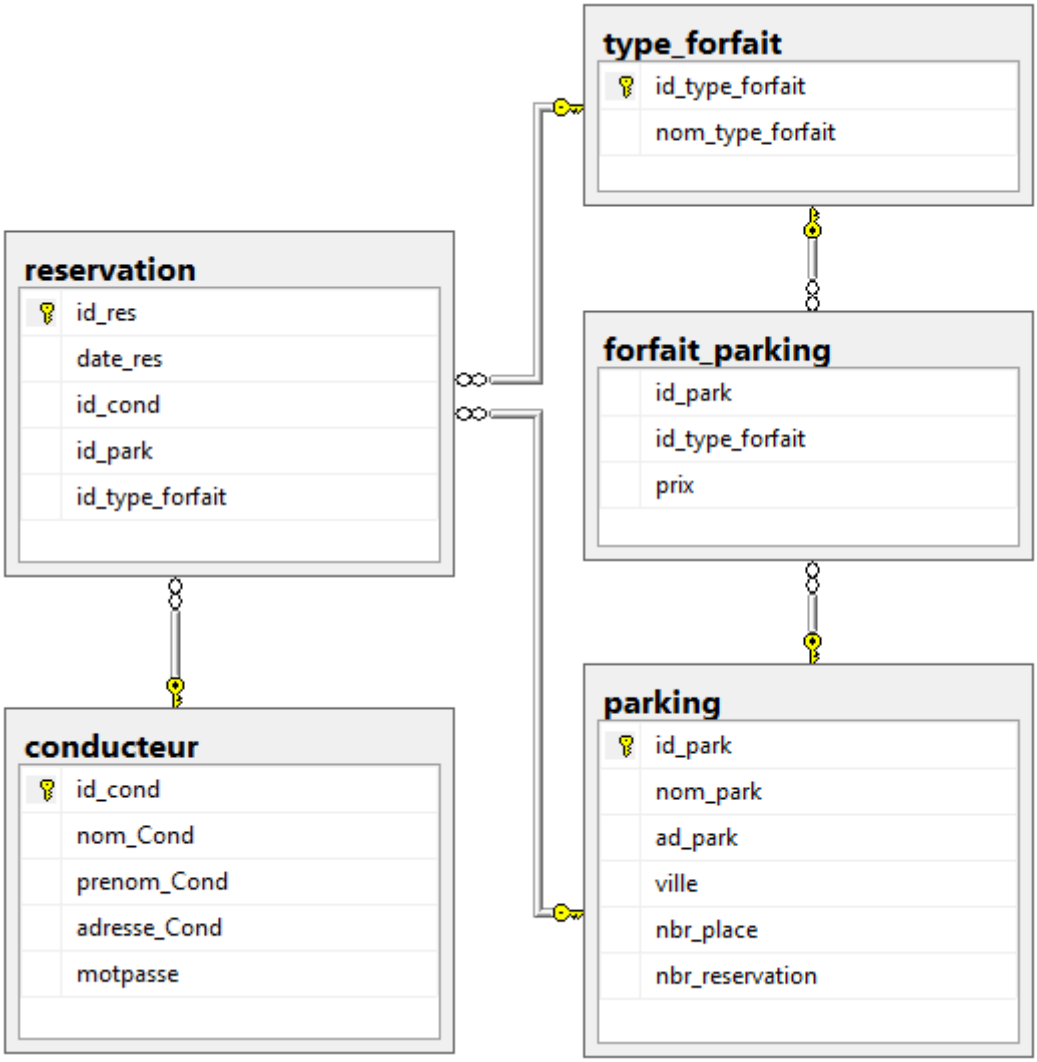
Agit en tant que médiateur entre l'objet DataSet et la base de données. Cela permet au dataset de contenir des données provenant de plusieurs bases de données ou d'autres sources de données. DataAdapter agit en tant que médiateur entre l'objet DataSet et la base de données. Cela permet au dataset de contenir des données provenant de plusieurs bases de données ou d'autres sources de données.

DataReader

C'est une alternative à la combinaison DataSet et DataAdapter. Cet objet fournit un accès orienté connexion aux enregistrements de données dans la base de données. Ces objets sont adaptés à un accès en lecture seule, tel que le remplissage d'une liste, puis la rupture de la connexion.

Exercice :

On veut réaliser un site web dynamique qui permet la gestion de parking.



Le site web doit être sécurisé à l'aide d'une page de connexion. Une fois un utilisateur est connecté, il redirigé vers une page d'accueil contenant un menu de navigation.

1- Réaliser une page de connexion permettant à un conducteur de se connecter au site web en fournissant son identifiant et son mot de passe.

2- Réaliser la page d'accueil avec le menu de navigation sous forme des liens hypertextes permettant d'atteindre les pages des questions qui suivent.

3- Créer une page d'ajout d'une réservation : l'identifiant du conducteur, du parking et du type de forfait sont choisis dans des listes déroulantes ; ajouter les validateurs pour valider les champs de saisis.

4- Créer une page web qui permet au conducteur connecté de rechercher un parking à réserver avec des places libres et qui ne dépasse pas un prix donné. Le prix est saisi dans une zone de texte. La liste affiche les noms des parkings sous forme de liens HyperText. Un clic sur le nom d'un parking, affiche les détails de ce parking : adresse, nombre de places libres, nom de type de forfait et prix.

5- Créer une page web qui permet à un conducteur connecté de consulter les dépenses concernant les réservations des parkings pendant une période donnée (entre deux dates). Une grille affiche la liste des réservations effectuées entre les deux dates avec les informations : nom des parkings, ville, prix.

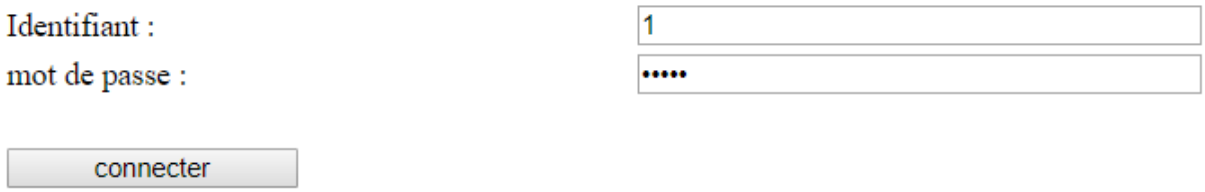
6- Afficher en bas de la grille une ligne pour afficher le total des réservations.
7- On veut réaliser un service web permettant de retourner le nombre de réservations effectuées pour un parking donné à une date donnée :
a) Développer ce service web.
b) Créer une page web permettant de tester ce service.

Corrigé :

La déclaration de connexion

```
SqlConnection cnx = new SqlConnection(@"server=BELABED;database=gestion_parking;integrated security=true;");
```

1)



Fichier Q1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Q1.aspx.cs" Inherits="efm2017.Q1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .auto-style1 {
            width: 100%;
        }
        .auto-style2 {
            width: 316px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <table class="auto-style1">
            <tr>
                <td class="auto-style2">
                    <asp:Label ID="Label1" runat="server" Text="Identifiant :"></asp:Label>
                </td>
                <td>
                    <asp:TextBox ID="TextBox1" runat="server" Width="282px"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td class="auto-style2">
                    <asp:Label ID="Label2" runat="server" Text="mot de passe :"></asp:Label>
                </td>
                <td>
                    <asp:TextBox ID="TextBox2" runat="server" Width="282px" TextMode="Password"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td class="auto-style2">
                    <asp:Label ID="Label3" runat="server" BackColor="#FF5050" BorderColor="#FF5050" Visible="False"></asp:Label>
                </td>
                <td> </td>
            </tr>
            <tr>
```

```
            <td class="auto-style2">
                <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="connecter" Width="147px"
/>
            </td>
            <td> </td>
        </tr>
    </table>
  </div>
  </form>
</body>
</html>
```

Fichier Q1.aspx.cs

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (TextBox1.Text != string.Empty && TextBox2.Text != string.Empty)
        {
            SqlDataAdapter da = new SqlDataAdapter("select * from conducteur where id_cond =@idcond and
motpasse=@pass", cnx);
            da.SelectCommand.Parameters.AddWithValue("@idcond", TextBox1.Text);
            da.SelectCommand.Parameters.AddWithValue("@pass", TextBox2.Text);
            DataTable dt = new DataTable();
            da.Fill(dt);
            if (dt.Rows.Count > 0)
            {
                Response.Redirect("menu.aspx");
                dt.Clear();
            }
            else
            {
                Label3.Text = "incorrect ";
                Label3.Visible = true;
            }
        }
    }
```

2)

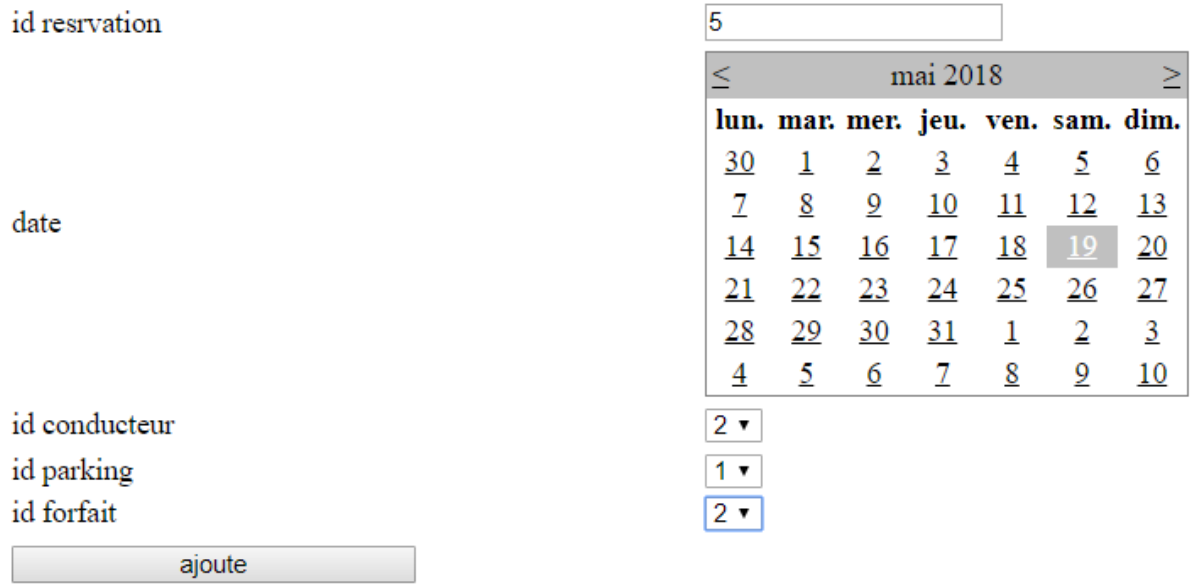
Question 1
Question 3
Question 4
Question 5

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="menu.aspx.cs" Inherits="efm2017.menu" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:Menu ID="Menu1" runat="server" BackColor="#FFFBD6" DynamicHorizontalOffset="2" Font-
Names="Verdana" Font-Size="0.8em" ForeColor="#990000" StaticSubMenuIndent="10px">
        <DynamicMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
        <DynamicMenuStyle BackColor="#FFFBD6" />
        <DynamicSelectedStyle BackColor="#FFCC66" />
        <Items>
            <asp:MenuItem Text="Question 1" Value="New Item" NavigateUrl="~/Q1.aspx"></asp:MenuItem>
            <asp:MenuItem Text="Question 3" Value="new item" NavigateUrl="~/Q3.aspx"></asp:MenuItem>
            <asp:MenuItem Text="Question 4" Value="new item" NavigateUrl="~/Q4.aspx"></asp:MenuItem>
            <asp:MenuItem Text="Question 5" Value="new item" NavigateUrl="~/Q5.aspx"></asp:MenuItem>
```

```
        </Items>
        <StaticHoverStyle BackColor="#990000" ForeColor="White" />
        <StaticMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
        <StaticSelectedStyle BackColor="#FFCC66" />
      </asp:Menu>

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

3)




```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Q3.aspx.cs" Inherits="efm2017.Q3" %>
<!DOCTYPE html>
<htmlxmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        .auto-style1 {
            width: 100%;
        }
        .auto-style2 {
            height: 23px;
        }
        .auto-style3 {
            height: 23px;
            width: 369px;
        }
        .auto-style4 {
            width: 369px;
        }
        .auto-style5 {
            width: 369px;
            height: 30px;
        }
        .auto-style6 {
            height: 30px;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
```

```
        <table class="auto-style1">
            <tr>
                <td class="auto-style3">id resrvation</td>
                <td class="auto-style2">
                    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td class="auto-style4">date</td>
                <td>
                    <asp:Calendar ID="Calendar1" runat="server" Height="16px"></asp:Calendar>
                </td>
            </tr>
            <tr>
                <td class="auto-style3">id conducteur</td>
                <td class="auto-style2">
                    <asp:DropDownList ID="DropDownList1" runat="server">
                    </asp:DropDownList>
                </td>
            </tr>
            <tr>
                <td class="auto-style4">id parking</td>
                <td>
                    <asp:DropDownList ID="DropDownList2" runat="server">
                    </asp:DropDownList>
                </td>
            </tr>
            <tr>
                <td class="auto-style4">id forfait</td>
                <td>
                    <asp:DropDownList ID="DropDownList3" runat="server">
                    </asp:DropDownList>
                </td>
            </tr>
            <tr>
                <td class="auto-style5">
                    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="ajoute" Width="217px" />
                </td>
                <td class="auto-style6"></td>
            </tr>
        </table>
    <div>
    </div>
    </form>
</body>
</html>
```

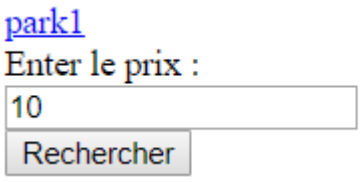
```
        protected void Page_Load(object sender, EventArgs e)
        {
            cnx.Open();

            SqlCommand cmd1 = new SqlCommand ("select id_cond from conducteur", cnx);
            SqlDataReader dr1 = cmd1.ExecuteReader();
            while (dr1.Read())
            {
                DropDownList1.Items.Add(dr1["id_cond"].ToString());
             DropDownList1.DataBind();
            }
            dr1.Close();

            SqlCommand cmd2 = new SqlCommand("select id_park from parking", cnx);
            SqlDataReader dr2 = cmd2.ExecuteReader();
            while (dr2.Read())
```

```
        {
            DropDownList2.Items.Add(dr2["id_park"].ToString());
            DropDownList2.DataBind();
        }
        dr2.Close();
        SqlCommand cmd3 = new SqlCommand("select id_type_forfait from type_forfait", cnx);
        SqlDataReader dr3 = cmd3.ExecuteReader();
        while (dr3.Read())
        {
            DropDownList3.Items.Add(dr3["id_type_forfait"].ToString());
            DropDownList3.DataBind();
        }
        dr3.Close();
        cnx.Close();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        cnx.Open();
        SqlCommand cmdA = new SqlCommand("insert into reservation values
(@idres,@dateres,@idcond,@idpark,@idtypeforfait)", cnx);
        cmdA.Parameters.AddWithValue("@idres", TextBox1.Text);
        cmdA.Parameters.AddWithValue("@dateres", Calendar1.SelectedDate.ToShortDateString());
        cmdA.Parameters.AddWithValue("@idcond", DropDownList1.Text);
        cmdA.Parameters.AddWithValue("@idpark", DropDownList2.Text);
        cmdA.Parameters.AddWithValue("@idtypeforfait", DropDownList3.Text);
        cmdA.ExecuteNonQuery();
        cnx.Close();
    }
```

4)



Fichier Q4.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Q4.aspx.cs" Inherits="efm2017.Q4" %>

<!DOCTYPE html>

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

        <asp:Label ID="Label1" runat="server" Text="Enter le prix :"></asp:Label>

    </div>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Rechercher" OnClick="Button1_Click" />
        <br />
    </form>
</body>
</html>
```

Fichier Q4.aspx.cs

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        SqlCommand cmd = new SqlCommand("select parking.nom_park, parking.id_park from parking,forfait_parking where parking.nbr_place>parking.nbr_reservation and forfait_parking.prix<=" + TextBox1.Text + " and parking.id_park=forfait_parking.id_park", cnx);
        cnx.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            Response.Write("<a href='detail_park.aspx?id="+dr[1]+"'>"+dr[0].ToString()+"</a><br/>");
        }
        dr.Close();
        cnx.Close();
    }
```

ad_park	nbr_place	nom_type_forfait	prix
centre ville	20	forfait2	5

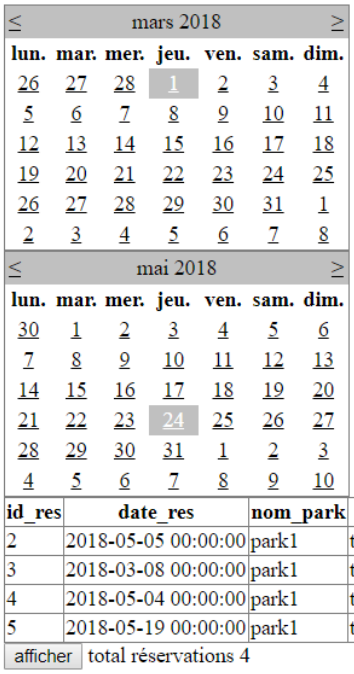
Fichier detail-park.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="detail_park.aspx.cs" Inherits="efm2017.detail_park" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
<asp:GridView ID="GridView1" runat="server">
    </asp:GridView>
</div>
  </form>
</body></html>
```

Fichier Detail.aspx.cs

```
    protected void Page_Load(object sender, EventArgs e)
    {
        string k = Request.QueryString["id"].ToString();
        SqlDataAdapter da = new SqlDataAdapter("SELECT parking.ad_park,parking.nbr_place,type_forfait.nom_type_forfait,forfait_parking.prix FROM parking INNER JOIN forfait_parking ON parking.id_park = forfait_parking.id_park INNER JOIN type_forfait ON forfait_parking.id_type_forfait = type_forfait.id_type_forfait where parking.id_park=@id", cnx);
        da.SelectCommand.Parameters.AddWithValue("@id",k);
        DataTable dt = new DataTable();
        da.Fill(dt);
        GridView1.DataSource = dt;
        GridView1.DataBind();
        cnx.Close();
    }
```

5) et 6)



≤		mars 2018				≥
lun.	mar.	mer.	jeu.	ven.	sam.	dim.
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

≤		mai 2018				≥
lun.	mar.	mer.	jeu.	ven.	sam.	dim.
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

id_res	date_res	nom_park	ville	prix
2	2018-05-05 00:00:00	park1	taourirt	5
3	2018-03-08 00:00:00	park1	taourirt	5
4	2018-05-04 00:00:00	park1	taourirt	5
5	2018-05-19 00:00:00	park1	taourirt	5

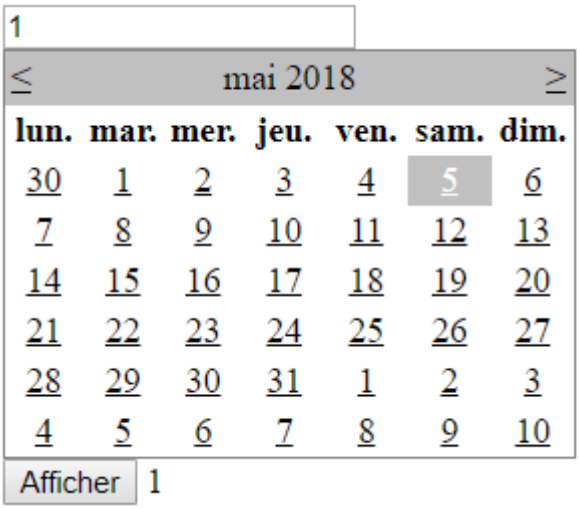
afficher total réservations 4

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Q5.aspx.cs" Inherits="efm2017.Q5" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
      <asp:Calendar ID="Calendar2" runat="server"></asp:Calendar>
      <asp:GridView ID="GridView1" runat="server">
      </asp:GridView>
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="afficher" />
      <asp:Label ID="Label1" runat="server"></asp:Label>
      </div>
   </form>
</body>
</html>
```

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        SqlDataAdapter da= new SqlDataAdapter("SELECT reservation.id_res,reservation.date_res, parking.nom_park,parking.ville,forfait_parking.prix FROM parking INNER JOIN reservation ON parking.id_park = reservation.id_park INNER JOIN forfait_parking ON parking.id_park = forfait_parking.id_park where reservation.date_res between '" + Calendar1.SelectedDate.ToShortDateString() + "' and '" + Calendar2.SelectedDate.ToShortDateString() + "'", cnx);
        DataTable dt = new DataTable();
        da.Fill(dt);
        GridView1.DataSource = dt;
        GridView1.DataBind();
        Label1.Text += "total réservations "+dt.Rows.Count.ToString();

    }
```

7)



Fichier WebService1.asmx.cs

```
    [WebMethod]
     public int nombre_reservation(int id,DateTime date_r)
     {
       cnx.Open();
       SqlCommand cmd=new SqlCommand("select * from reservation where id_park="+id+"and date_res='"+date_r+"'",cnx);
       SqlDataReader dr = cmd.ExecuteReader();
       int k = 0;
       while(dr.Read())
       {
          k++;
       }
       dr.Close();
       cnx.Close();
       return k;

     }

   internal string nombre_reservation(string p1, string p2)
   {
      throw new NotImplementedException();
   }
```

Fichier web_service.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="web_s.aspx.cs" Inherits="efm2017.web_s" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
      <asp:Button ID="Button1" runat="server" Text="Afficher" OnClick="Button1_Click" />
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
   </div>
   </form>
</body>
</html>
```

Fichier web_s.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
   WebService1 ws = new WebService1();
   Label1.Text =ws.nombre_reservation(int.Parse(TextBox1.Text),Calendar1.SelectedDate).ToString();
}
```

ASP.NET MVC

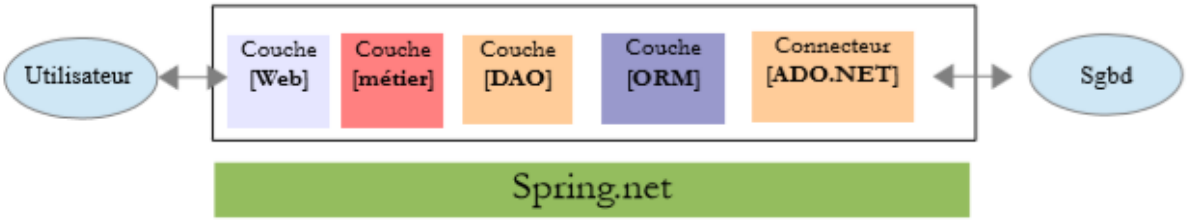
Un Framework Web .NET qui fournit un cadre pour développer des applications Web selon le modèle MVC (Modèle – Vue – Contrôleur).

Modèle : représente la forme des données et le logique métier. Il maintient les données d'application.Les objets de modèle récupèrent et stockent l'état du modèle dans une base de données.

Vue : est une interface utilisateur. Afficher les données d'affichage en utilisant le modèle pour l'utilisateur et leur permet également de modifier les données.

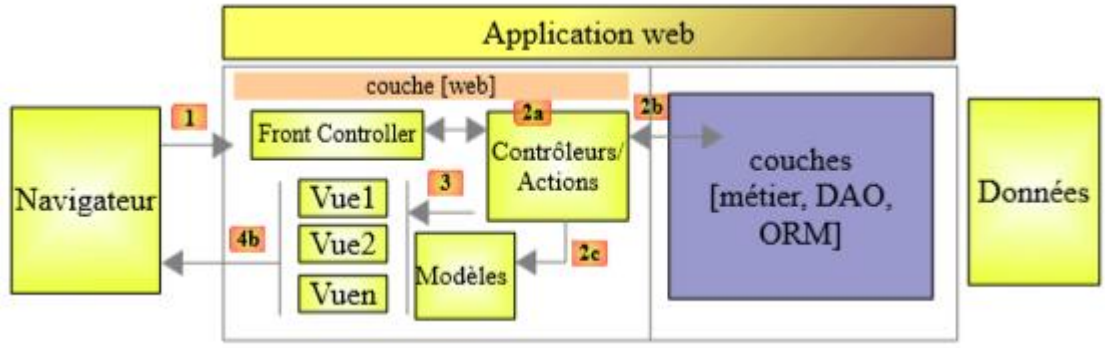
Contrôleur : gère la demande de l'utilisateur .Généralement, l'utilisateur interagit avec View, ce qui génère une requête sera gérée par un contrôleur. Le contrôleur affiche la vue appropriée avec les données du modèle en tant que réponse.

La place d'ASP.NET MVC dans une application Web :



- la couche Web est la couche en contact avec l'utilisateur de l'application Web. Celui-ci interagit avec l'application Web au travers de pages Web visualisées par un navigateur. C'est dans cette couche que se situe ASP.NET MVC et uniquement dans cette couche ;
- la couche métier implémente les règles de gestion de l'application, tels que le calcul d'un salaire ou d'une facture. Cette couche utilise des données provenant de l'utilisateur via la couche [Web] et du SGBD via la couche [DAO].
- la couche DAO (Data Access Objects), la couche ORM (Object Relational Mapper) et le connecteur ADO.NET gèrent l'accès aux données du SGBD.
- la couche ORM fait un pont entre les objets manipulés par la couche [DAO] et les lignes et les colonnes des tables d'une base de données relationnelle. Deux ORM sont couramment utilisés dans le monde .NET, NHibernate et Entity Framework.
- l'intégration des couches peut être réalisée par un conteneur d'injection de dépendances (Dependency Injection Container) tel que Spring.

Le modèle de développement d'ASP.NET MVC



Le traitement d'une demande d'un client se déroule de la façon suivante :

- demande : les URL demandées sont de la forme http://machine:port/contexte/Controlleur/Action/param1/param2/....? p1=v1&p2=v2&...
  Le [Front Controller] utilise un fichier de configuration pour " router " la demande vers le bon contrôleur et la bonne action au sein de ce contrôleur. Pour cela, il utilise le chemin [Controlleur/Action] de l'URL. Le reste de l'URL [/param1/param2/...] sont des paramètres facultatifs qui seront transmis à l'action.
  Le C de MVC est ici la chaîne [Front Controller, Contrôleur, Action]. Si le chemin [Controlleur/Action] n'aboutit pas à un contrôleur existant ou une action existante, le serveur Web répondra que l'URL demandée n'a pas été trouvée.
- traitement : l'action choisie peut exploiter les paramètres parami que le [Front Controller] lui a transmis. Une fois la demande du client traitée, celle-ci peut appeler diverses réponses.
  Un exemple classique est : • une page d'erreur si la demande n'a pu être traitée correctement
  • une page de confirmation sinon
  • l'action demande à une certaine vue de s'afficher. Cette vue va afficher des données qu'on appelle le modèle de la vue.
- réponse : la vue V choisie utilise le modèle M construit par l'action pour initialiser les parties dynamiques de la réponse HTML qu'elle doit envoyer au client puis envoie cette réponse.

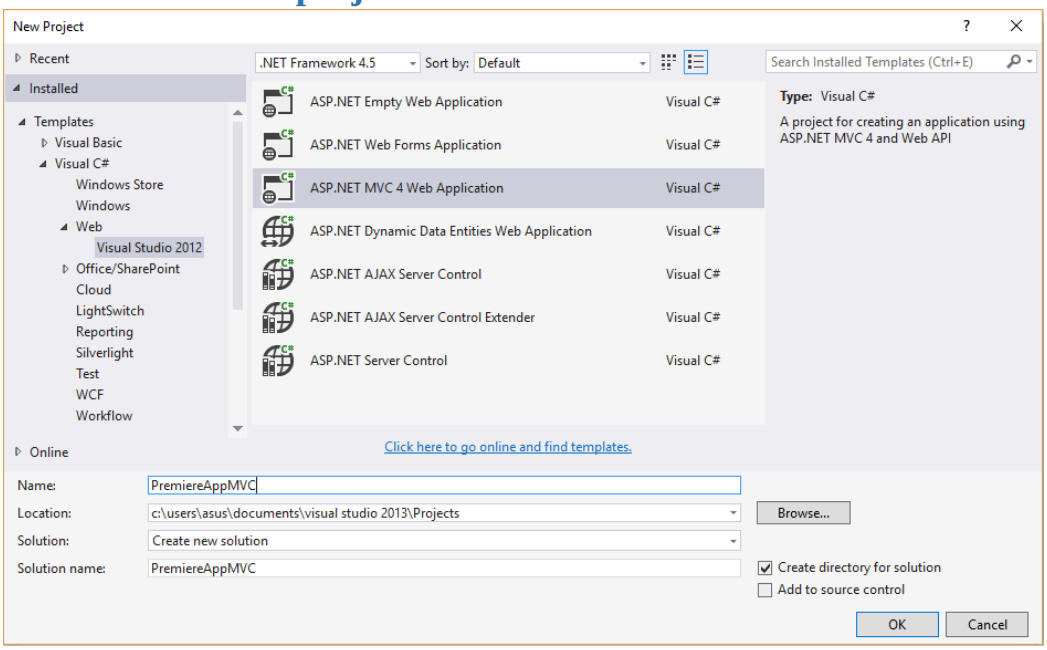
Envoi à un serveur Web par un client Web des valeurs d'un formulaire

Un client Web peut utiliser deux méthodes différentes appelées POST et GET pour envoyer des données à un serveur web. L'attribut method="méthode", avec méthode égal à GET ou POST, de la balise <form> indique au navigateur la méthode à utiliser pour envoyer les informations recueillies dans le formulaire à l'URL précisée par l'attribut action="URL". Lorsque l'attribut method n'est pas précisé, c'est la méthode GET qui est prise par défaut.

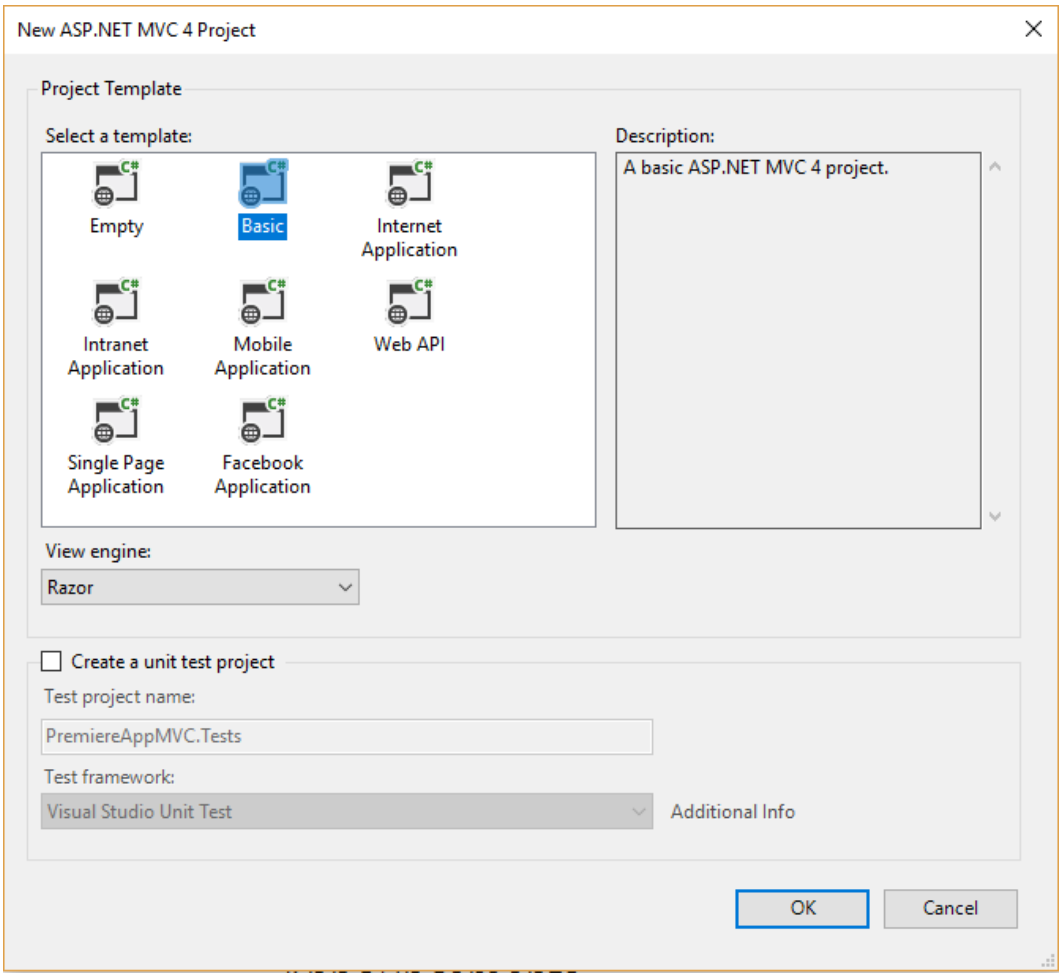
La méthode GET : envois toutes les informations collectées dans le cadre de l'URL.

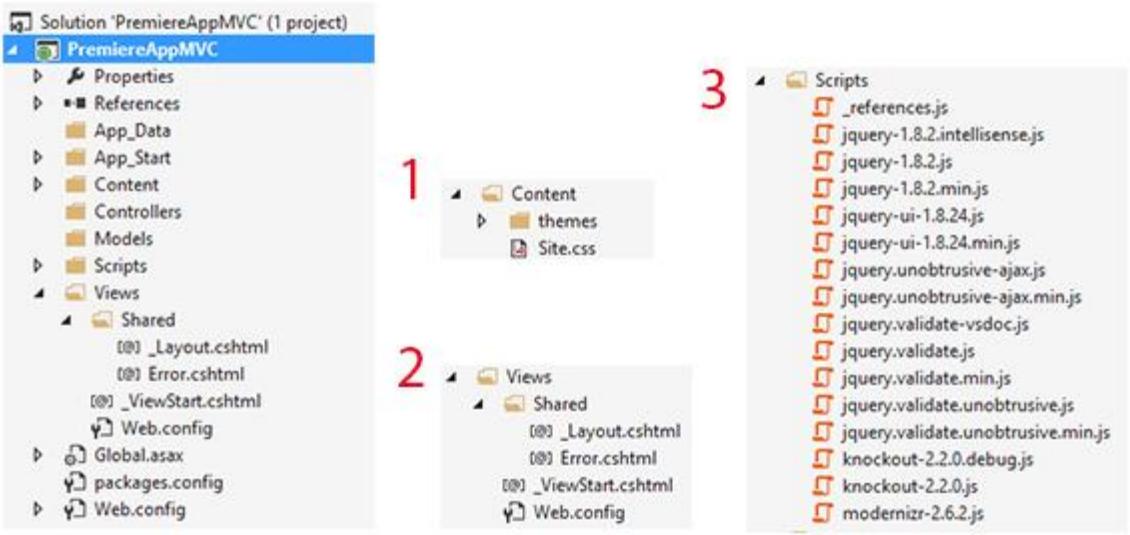
La méthode POST : transmet l'information de manière invisible à l'utilisateur.

La structure d'un projet ASP.NET MVC



Nous choisissons un projet ASP.NET MVC de base. Ce modèle nous fournit une application Web vide avec cependant toutes les ressources (DLL, bibliothèques javascript, ...) pour travailler.





l'architecture du projet reflète son modèle MVC :

• les contrôleurs C seront placés dans le dossier [Controllers],

• les modèles de données M seront placés dans le dossier [Models],

• les vues V seront placées dans le dossier [Views],

• en [1], le fichier [Site.css] sera le fichier CSS par défaut de l'application ;

• en [2], on trouve trois vues particulières : _ViewStart, _Layout, Error ;

• en [3], un certain nombre de bibliothèques Javascript sont mises à notre disposition.

Le fichier _ViewStart est le suivant :

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

ligne 1 : le caractère @ annonce une séquence C# dans la vue. En effet, on peut inclure du code C# dans une vue ;

ligne 2 : définit une variable Layout qui définit la vue parente de toutes les vues. Elle correspond à la page maître du framework ASP.NET classique.

Le fichier _Layout est le suivant :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
```

```
    @RenderBody()

    @Scripts.Render("~/bundles/jquery")
    @RenderSection("scripts", required: false)
</body>
</html>
```

Lorsqu'une vue du dossier [Views] sera rendue, son corps sera rendu par @RenderBody(). Cela signifie que la vue n'a pas à inclure les balises <html>, <head>, <body>. Celles-ci sont fournies par le fichier ci-dessus. Celui-ci est pour l'instant assez hermétique.

• [Web.config] est le fichier de configuration de l'application Web. Il est complexe. On sera amené à le modifier lorsqu'on utilisera le framework [Spring.net] dans une architecture multicouche. • [Global.asax] contient le code exécuté au démarrage de l'application. En général, ce code exploite les différents fichiers de configuration de l'application dont [Web.config].

++++++++++++++++ Bon courage ++++++++++++++++