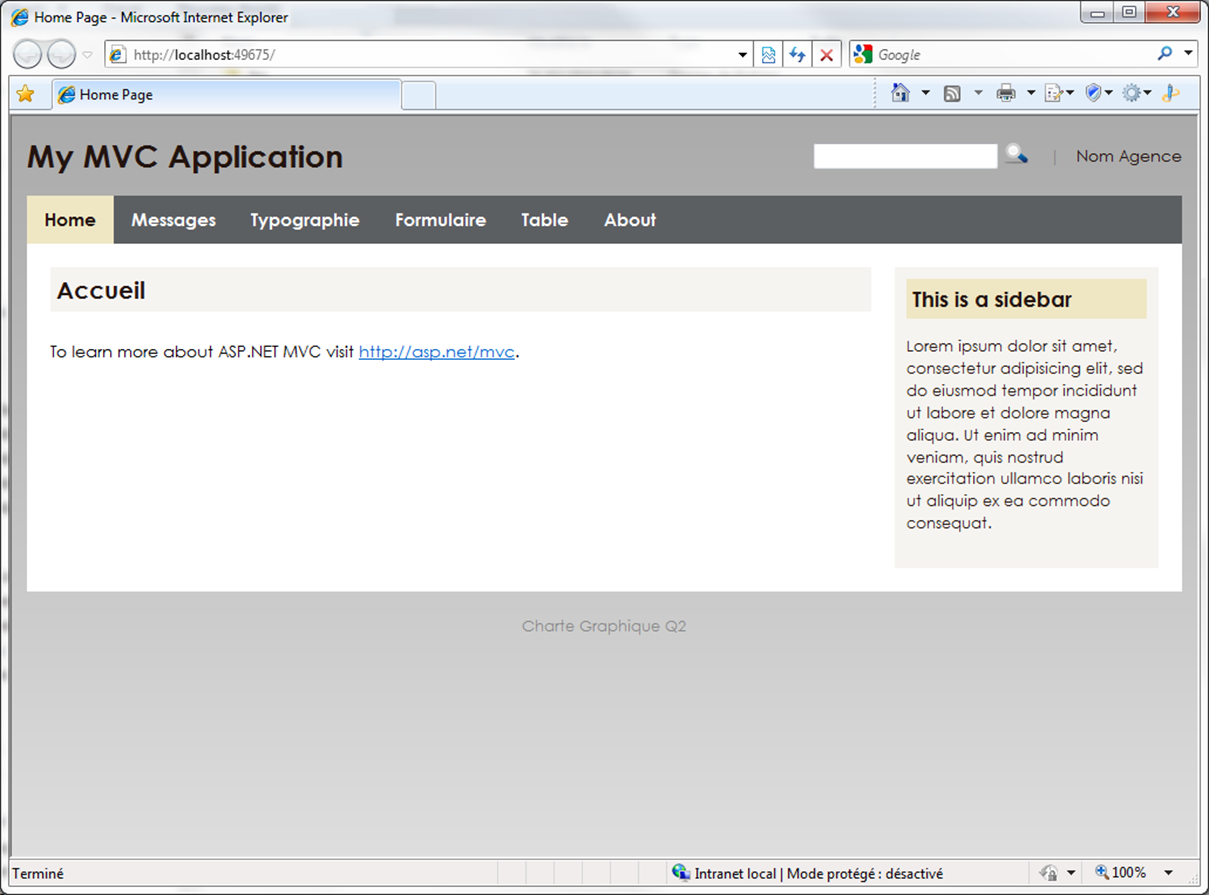
# Q2 / Charte Graphique

## Présentation



### Disposition générale de la page

* Un bandeau de titre qui comprend :
  + Un « titre » qui sert de lien pour revenir à la page d’accueil
  + Une mini barre d’outils pour faire une recherche rapide et changer d’agence
* Une barre de menu pour présenter les « grosses » fonctions de l’application
* Un espace de contenu découpé en deux parties :
  + Le contenu principal sur la gauche
  + Un bandeau latéral sur la droite
* Un pied de page

## Le fichier Site.Master

### Contenu simplifié

<%@ Master Language="C#" Inherits="System.Web.Mvc.ViewMasterPage" %>

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

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<link href="<%= Url.Content("~/Content/style.css") %>" rel="stylesheet" type="text/css" />

<script src="<%= Url.Content("~/Scripts/jquery-1.3.2.js") %>" type="text/javascript"></script>

**<asp:ContentPlaceHolder ID="Htmlhead" runat="server" />**

</head>

<body id="site">

<div id="page">

<div id="*header*" class="clearfix">

<h1><a href="#">My MVC Application</a></h1>

<div>

<%= Html.TextBox("s") %>

<a href="#" class="search" title="Lancer la recherche"></a>

<span> | </span>

<a href="#" title="Changer d'agence">Nom Agence</a>

</div>

</div>

<ul id="*nav*" class="clearfix">

<li><%= Html.ActionMenu("Home", "Index", "Home") %></li>

<li><%= Html.ActionMenu("About", "About", "Home")%></li>

</ul>

<div id="*main*" class="clearfix">

<div id="content">

**<asp:ContentPlaceHolder ID="Content" runat="server" />**

</div>

<div id="sidebar">

**<asp:ContentPlaceHolder ID="Sidebar" runat="server" />**

</div>

</div>

<div id="*footer*" class="clearfix">

Charte Graphique Q2

</div>

</div>

</body>

</html>

### Les PlaceHolders

Ce fichier contient 3 contrôles PlaceHolder à utiliser dans les vues :

* Htmlhead : pour ajouter du contenu HTML dans la section <head> de la page
* Content : pour définir le contenu HTML principal de la page
* Sidebar : pour afficher du contenu HTML dans la barre latérale

Lorsque la colonne « Sidebar » est vide, une fonction jQuery élargi automatiquement la zone « Content » pour qu’elle utilise toute la largeur disponible.

Attention, il ne s’agit pas des PlaceHolders définis par défaut à la création d’une application MVC :

* TitleContent : pour indiquer le titre de la page (sera inclus dans une balise <title>)
* MainContent : pour définir le contenu HTML de la page

L’avantage, c’est que le PlaceHolder « Htmlhead » peut servir pour plus de contenu que le simple titre de la page. On peut ainsi y faire apparaitre les feuilles de style et les javascripts spécifiques aux différentes vues :

<asp:Content ID="Htmlhead" ContentPlaceHolderID="Htmlhead" runat="server">

**<title>**Table - Consultation**</title>**

<script src="/jQuery/jquery.fitted.js" type="text/javascript"></script>

<script type="text/javascript">

$(document).ready(function() {

$('tr').fitted();

});

</script>

<style type="text/css">

.hovered

{

cursor: pointer;

}

</style>

</asp:Content>

### Fichiers référencés

Par ailleurs, ce fichier Site.master référence deux fichiers externes :

* /Content/style.css : la feuille de style principale de Q2
* /Scripts/jquery-version.js : la librairie jQuery

Lorsque la charte graphique sera utilisée dans le cadre d’une application particulière, il faudra aussi référencer une seconde feuille de styles spécifique à cette application. Par exemple :

<head>

<link href="<%= Url.Content("~/Content/style.css") %>" rel="stylesheet" type="text/css" />

<link href="<%= Url.Content("**~/Content/QAgence/QAgence.css**") %>" rel="stylesheet" type="text/css" />

...

De cette façon, le fichier « ~/Content/style.css » ne sera pas encombré avec les styles spécifiques aux différentes applications et sera facilement échangeable par les différentes applications

## Les éléments typographiques

### A l’intérieur du PlaceHolder « Content »

* <h2> pour le titre principal
* <h3> pour les sous-titres
* <div class="xxxxxbox"> pour les zones de message
* <p> pour les autres contenus

### A l’intérieur du PlaceHolder « Sidebar »

* <div class="box"> pour englober une zone logique de contenu
* <h3> pour les titres
* <ul class="buttons"> pour les listes de menu

### Les classes pour les zones de message

* messagebox : attirer l'attention de l'utilisateur et faire ressortir une zone particulière sur la page
* infobox : envoyer un message d'information à l'utilisateur en relation avec le contenu de la page en cours
* successbox : annoncer à l'utilisateur que l'action qu'il vient de réaliser s'est déroulée correctement
* warningbox : avertir l'utilisateur que l'action qu'il vient de lancer a été exécutée bien qu'elle pose quelques soucis
* errorbox : signaler à l'utilisateur qu'il y a un problème bloquant avec l'action qu'il vient d'exécuter ou qu'il veut exécuter

## Les boutons d’action

Il s’agit des boutons qui vont servir à appeler des actions pour consulter le détail d’une fiche, ajouter un nouvel enregistrement, modifier un contenu… Ils sont à créer sous forme de liens (balise <a>) en définissant un nom de classe spécifique à l’action visée :

* <a class="show" ...> : consulter
* <a class="new" ...> : ajouter
* <a class="edit" ...> : modifier
* <a class="delete" ...> : supprimer
* <a class="search" ...> : chercher
* <a class="next" ...> : suite / aller vers quelque chose
* <a class="back" ...> : retour / sortir de quelque chose

### Cas des actions imprévues

Lorsqu’il est nécessaire de définir un bouton d’action différent, il faut utiliser la classe « icon » plus une classe supplémentaire spécifique à ce type de bouton :

<a class="icon bloquer" ...>

Puis il faut définir la classe dans la feuille de style de l’application concernée pour indiquer quelle icône utiliser :

a.bloquer

{

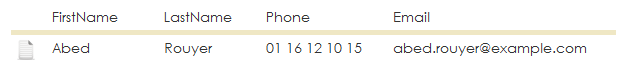
background-image: url(bloquer.png);

}

### Utilisations standards

Par défaut, un bouton d’action est prévu pour s’utiliser tel quel. Il n’est pas nécessaire de définir un contenu texte pour afficher dans le lien.

Par exemple, dans une table :



<tr>

<td>**<a class="show" href="#"></a>**</td>

<td>Abed</td>

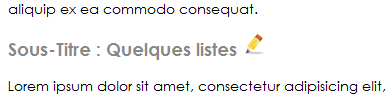
<td>Rouyer</td>

<td>01 16 12 10 15</td>

<td>abed.rouyer@example.com</td>

</tr>

Ou à la suite d’un sous-titre :



<h3>

Sous-Titre : Quelques listes

**<a class="edit" href="#"></a>**

</h3>

### Ajouter un bouton sur un titre

Quand on veut faire apparaitre un bouton tout à droite d’un titre, il est nécessaire d’intégrer le lien à l’intérieur d’une balise <span> :

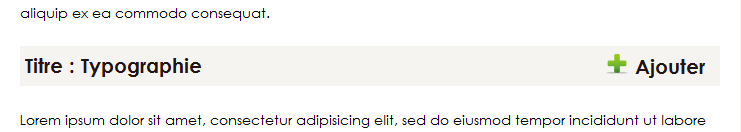
<h2>

Titre : Typographie

<span>**<a class="new" href="#">Ajouter</a>**</span>

</h2>

Ce qui donne :



### Ajouter un bouton dans la sidebar

Pour ajouter un « gros » bouton dans la barre latérale, il faut exceptionnellement utiliser une balise <h2> à l’intérieur du PlaceHolder « Sidebar » :

<div class="box">

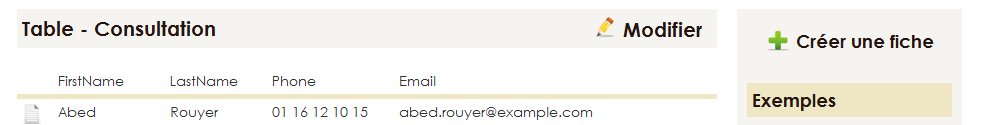
<h2>

**<a class="new" href="/Home/New">Créer une fiche</a>**

</h2>

</div>

Ce qui donne :



## Les boutons de menu

Pour créer des menus verticaux dans la barre latérale, il faut passer par une balise <ul> à laquelle on ajoute la classe « buttons » :

<div class="box">

<h3>Exemples</h3>

**<ul class="buttons">**

<li><a href="#">Consultation</a></li>

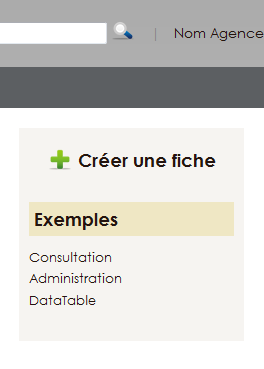
<li><a href="#">Administration</a></li>

<li><a href="#">DataTable</a></li>

</ul>

</div>

Ce qui donne :



## Utiliser des tables

Quand on souhaite utiliser une table, il faut respecter les deux règles suivantes :

* Ajouter la classe « xtable » à la balise <table>
* Définir du code HTML valide (avec au moins un élément <thead> et un élément <tbody>)

Exemple minimum de code HTML pour une table :

<table class="xtable">

<thead>

<tr>

<th>&nbsp;</th>

<th>FirstName</th>

<th>LastName</th>

<th>Phone</th>

<th>Email</th>

</tr>

</thead>

<tbody>

<tr>

<td><a class="show" href="#"></a></td>

<td>Abed</td>

<td>Rouyer</td>

<td>01 16 12 10 15</td>

<td>abed.rouyer@example.com</td>

</tr>

...

</tbody>

</table>

Par défaut, le contenu à l’intérieur des tables s’affiche avec taille de police de caractères qui est légèrement diminuée par rapport au reste du texte. Dans le cas où le contenu de la table est important (et qu’il n’est pas trop dense), il est possible de rétablir la taille normale en ajoutant la classe « bigfont » à la balise <table> :

<table class="xtable bigfont">

...

## Afficher une fiche

Cette action correspond aux vues de type « Détail » qui sont destinées à afficher le contenu d’un enregistrement. Pour cela, il est possible d’utiliser un code HTML basique qui permet d’obtenir une présentation homogène des informations :

*<h2>*

*Contact*

*<span><a class="edit" href="#">Modifier</a></span>*

*</h2>*

**<fieldset class="form">**

<p>

<label>Prénom</label>

<span>Pierre</span>

</p>

**<p>**

**<label>**Nom</label>

<span>Azerty</span>

</p>

<p>

<label>Téléphone</label>

<span>01 23 45 67 89</span>

</p>

<p>

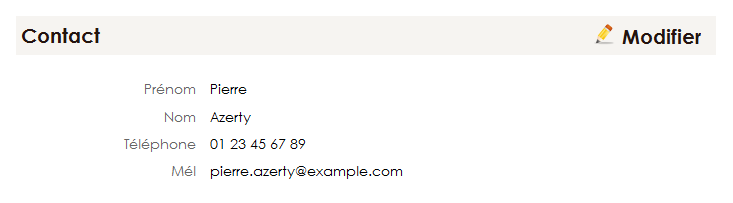
<label>Mél</label>

<span>pierre.azerty@example.com</span>

</p>

</fieldset>

Ce qui donne :



## Créer un formulaire

Par rapport à une simple vue « Détail », un formulaire doit aussi permettre de modifier les informations et de les envoyer par POST vers l’action du contrôleur chargée de gérer l’enregistrement de ces données.

Par conséquent, un formulaire doit au minimum contenir :

* Des balises <input> pour effectuer la saisie des données,
* Une balise <input type="submit"> pour poster les données,
* Un lien pour abandonner la saisie.

Le code HTML pour effectuer tout cela est construit en reprenant le même principe que pour l’affichage des données, c'est-à-dire en suivant le même enchainement de balises <fieldset>, <p> et <label> :

<h2>Modification</h2>

<%= Html.ValidationSummary("Merci de corriger avant de revalider.") %>

<% using (Html.BeginForm()) {%>

<fieldset class="form">

<p>

<label for="FirstName">Prénom</label>

<%= Html.TextBox("FirstName")%>

<%= Html.ValidationMessage("FirstName", "\*")%>

</p>

<p>

<label for="LastName">Nom</label>

<%= Html.TextBox("LastName")%>

<%= Html.ValidationMessage("LastName", "\*")%>

</p>

<p>

<label for="Phone">Téléphone</label>

<%= Html.TextBox("Phone", ViewData["Phone"])%>

<%= Html.ValidationMessage("Phone", "\*")%>

</p>

<p>

<label for="Email">Mél</label>

<%= Html.TextBox("Email", ViewData["Email"])%>

<%= Html.ValidationMessage("Email", "\*")%>

</p>

</fieldset>

<fieldset class="formsubmit">

<p>

<input type="submit" value="OK" />

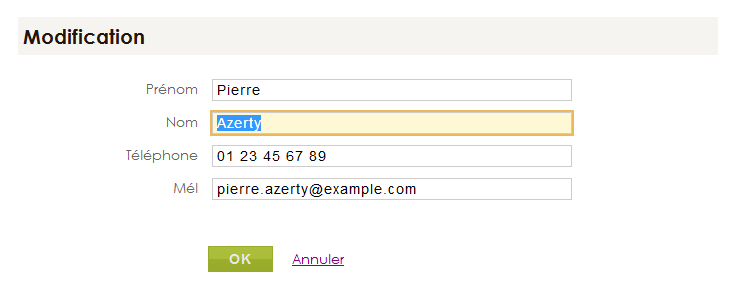
<%= Html.ActionLink("Annuler", "Index")%>

</p>

</fieldset>

<% } %>

Ce qui donne le résultat suivant :



Le fait d’utiliser la syntaxe « <label for="#"> » permet d’accéder automatiquement à la zone de saisie suivant le label lorsqu’on clique sur le libellé du label.

## Des formulaires plus élaborés

### Aérer le formulaire

Il est possible de définir plusieurs balises <fieldset> à l’intérieur d’un même formulaire. Cela permet entre autre de mieux organiser la structure de la saisie.

Dans ce cas, il faut compléter les balises <fieldset> par des balises <legend>, ce qui permet de définir un titre pour à chaque regroupement de données.

Le code dans la vue :

<fieldset class="form">

**<legend>Identité</legend>**

<p>

<label>Civilité</label>

...

</p>

<p>

<label for="Nom">Nom</label>

...

</p>

<p>

<label for="Prenom">Prénom</label>

...

</p>

</fieldset>

<fieldset class="form">

**<legend>Coordonnées</legend>**

<p>

<label for="Adresse1">Adresse</label>

...

</p>

Ce qui donne :



### Liste de boutons radio

Le code dans la vue :

<p>

<label>Civilité</label>

<%= Html.RadioButton("Civilite", "MR", true, new { @id = "Monsieur" }) %>

<label for="Monsieur">Monsieur</label>

<%= Html.RadioButton("Civilite", "MME", false, new { @id = "Madame" }) %>

<label for="Madame">Madame</label>

<%= Html.RadioButton("Civilite", "MLE", false,

new { @id = "Mademoiselle" }) %>

<label for="Mademoiselle">Mademoiselle</label>

</p>

Le code HTML généré :

<p>

<label>Civilité</label>

<input checked="checked"

id="Monsieur" name="Civilite" type="radio" value="MR" />

<label for="Monsieur">Monsieur</label>

<input id="Madame" name="Civilite" type="radio" value="MME" />

<label for="Madame">Madame</label>

<input id="Mademoiselle" name="Civilite" type="radio" value="MLE" />

<label for="Mademoiselle">Mademoiselle</label>

</p>

Ce qui donne :



Remarque :

* La ligne « <label>Civilité</label> » n’utilise pas la forme « <label for="#"> » parce que le fait de cliquer sur « Civilité » ne doit pas provoquer la sélection d’une des 3 civilités proposées.
* L’objet « new { @id = "Monsieur" }) » permet de générer un attribut « id » différent de l’attribut « name » et ainsi de sélectionner le bon bouton de radio quand on clique sur le libellé associé.

### Liste de cases à cocher

Le code dans la vue :

<p>

<label>Contacter par</label>

<%= Html.CheckBox("ContactTel")%><label for="ContactTel">Téléphone</label>

<%= Html.CheckBox("ContactMel")%><label for="ContactMel">Email</label>

<%= Html.CheckBox("ContactSms")%><label for="ContactSms">SMS</label>

</p>

Le code HTML généré :

<p>

<label>Contacter par</label>

<input id="ContactTel" name="ContactTel" type="checkbox" value="true" />

<input name="ContactTel" type="hidden" value="false" />

<label for="ContactTel">Téléphone</label>

<input id="ContactMel" name="ContactMel" type="checkbox" value="true" />

<input name="ContactMel" type="hidden" value="false" />

<label for="ContactMel">Email</label>

<input id="ContactSms" name="ContactSms" type="checkbox" value="true" />

<input name="ContactSms" type="hidden" value="false" />

<label for="ContactSms">SMS</label>

</p>

Ce qui donne :



Remarque :

* La ligne « <label>Contacter par</label> » n’utilise pas la forme « <label for="#"> » parce que le fait de cliquer sur « Contacter par » ne doit pas provoquer la sélection d’une des 3 cases à cocher.
* Pour chaque case à cocher, MVC génère une seconde case à cocher masquée qui lui sert à mémoriser la valeur actuelle de la case à cocher.
* Il n’est pas nécessaire d’utiliser un objet « new { @id = "xxx" }) » parce que chaque case à cocher est générée avec un attribut « id » spécifique.

### Liste verticale pour les boutons radio ou les cases à cocher

Le code dans la vue :

<p>

<label>CheckBoxes</label>

<%= Html.CheckBox("CheckBox1")%>

<label for="CheckBox1">CheckBox 1</label>

**<br /><label class="spacer"></label>**

<%= Html.CheckBox("CheckBox2", false)%>

<label for="CheckBox2">CheckBox 2</label>

<br /><label class="spacer"></label>

<%= Html.CheckBox("CheckBox3", false)%>

<label for="CheckBox3">CheckBox 3</label>

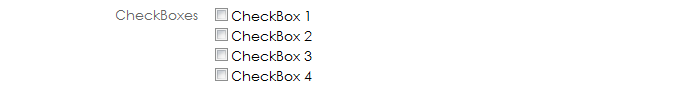
<br /><label class="spacer"></label>

<%= Html.CheckBox("CheckBox4", false)%>

<label for="CheckBox4">CheckBox 4</label>

</p>

Ce qui donne :



### Gérer deux zones de saisie contiguës verticalement

Le code dans la vue :

<p>

<label for="Adresse1">Adresse</label>

<%= Html.TextBox("Adresse1", "")%>

**<br /><label class="spacer"></label>**

<%= Html.TextBox("Adresse2", "")%>

</p>

Ce qui donne :



L’interligne entre les deux zones est moins importante que celle qui existe entre deux zones de saisie placées dans deux balises <p> différentes.

### Gérer deux zones de saisie contiguës horizontalement

Le code dans la vue :

<p>

<label for="CodePostal">CP / Ville</label>

<%= Html.TextBox("CodePostal", "", new { @type = "number" })%>

<%= Html.TextBox("Ville")%>

</p>

Ce qui donne :

