



Module 2 Développer pages web en lien avec base de données

Programmation Server Side	Séance	S03	Activité	A-001
---------------------------	--------	-----	----------	-------

Coder un composant serveur ayant pour objet de récupérer les valeurs saisies par l'utilisateur sur un formulaire.

Vous allez vous familiariser durant cette première activité avec les objets intrinsèques d'ASP net que sont les objets HttpResponse et HttpRequest.

Vous allez apprendre à manipuler les collections de paires clés/valeurs.

Sommaire de l'activité proposée :

Comprendre les mécanismes de traitement des formulaires	2
1.1 Amendement de la page Formulaire Client	2
1.2 Codification de la page formulaireClient.aspx	3
1.3 Modification pour prendre en compte la méthode POST	5

Comprendre les mécanismes de traitement des formulaires

Vous allez réaliser ici votre première page asp.net.
Objectif : récupérer les valeurs transmises via un formulaire.

1.1 Amendement de la page Formulaire Client

Le premier travail à réaliser consiste à programmer la soumission des éléments de formulaires et à demander l'exécution de l'action définie comme attribut du formulaire en recourant à la commande http GET.

Récupérez le document source sur le site ou utiliser le formulaire créer initialement lors de l'atelier de création de scripts JS.

Prenez connaissance du formulaire. Il présente la quasi-totalité des contrôles HTML de formulaire (Bouton Radio, Cases à cocher, Liste d'options, Boîte de texte).

Ce formulaire propose quelques illustrations de manipulation des éléments de formulaire avec Javascript et quelques fonctionnalités de modification dynamique de styles avec le DOM.
Toutefois, ces techniques n'ont maintenant plus de secret pour vous...

Intéressons-nous plus particulièrement à la balise **form** :

Un zoom sur les attributs de la balise form :

- L'attribut **action** désigne la page devant traiter la requête du client. Il peut s'agir de la page elle-même ou de tout autre page. Il s'agit ici de la page asp qui devra traiter le formulaire. Nous pourrions aussi, dans une autre architecture logicielle, disposer d'un script php ou d'un composant CGI.
- L'attribut **method** précise comment sont transmises les valeurs du formulaire. Deux valeurs possibles **get** et **post** qui correspondent respectivement à la commande du protocole http exécutée.
 - La valeur **Get** précise que les éléments sont transmis dans l'url. L'appel de la méthode **submit()** déclenchera donc une requête similaire à celle-ci :
http://localhost:1846/A002A.aspx?rdSexe=H&txtNom=d&txtPrenom=d&txtAdresse_P1=d&txtAdresse_P2=&txtCodePostal=19270
 - La valeur **post** précise que les valeurs des éléments de formulaire sont stockées dans le corps de la requête http.
- L'attribut **enctype** spécifie le codage des données transmises (le type mime). Cette valeur est attribuée par défaut à `"application/x-www-form-urlencoded"`. Le type MIME (Multipurpose Internet Mail Extensions) est un standard défini pour étendre les possibilités du mail, d'où son acronyme. Il permet ainsi de transmettre des documents de différents types (image, vidéo, texte, PostScript, ...) dans un courrier. Il permet dans le cadre d'échanges http d'indiquer le type de document transmis par le serveur à un client afin que ce dernier sache de quelle façon le traiter. Dans le cadre d'une transmission des éléments d'un formulaire client, en dehors du type indiqué ci-dessus, il est possible d'utiliser le codage `"multipart/form-data"` pour uploader des données. Cette approche est toutefois aujourd'hui désuète : nous utiliserons pour cela des classes spécialisées.

Travail à réaliser :

Modifiez la balise **form** pour indiquer que les valeurs du formulaire doivent être transmises dans l'URL. Indiquez au niveau de l'attribut **action** le nom de la page qui devra traiter les données, ici **FormulaireClient** (Ne pas préciser l'extension pour éviter des problèmes de routage)
Sur l'événement click du bouton btnValider, soumettez les données du formulaire **si aucune erreur ne subsiste** par appel de la méthode submit() de l'objet formulaire.

1.2 Codification de la page formulaireClient.aspx

Vous allez coder cette page dans le flux à l'aide des délimiteurs `<% %>`.

Ce n'est pas l'approche la plus pertinente mais elle vous permet ici de découvrir des objets Asp.Net fondamentaux avant d'aborder les contrôles Web.

Elle a pour principal inconvénient de mélanger allègrement blocs html et blocs de scripts client ou serveur, éléments d'interface et code d'implémentation des fonctions métier.

Objectif : Afficher dans un tableau HTML les noms des éléments de formulaire et leurs valeurs. Se familiariser avec les collections de paires clés/valeurs.

Vous allez principalement utiliser 2 classes représentant des objets ASP.Net intégrés:

- La classe **HttpRequest** : Correspond à la requête **transmise par** le client
- La classe **HttpResponse** : Correspond à la requête transmise au client **en réponse** à sa demande

Vous allez récupérer les données transmises via le formulaire dans une collection de type **NameValueCollection**. Ce type représente une collection de paires clés/valeurs où les clés et les valeurs sont de type string. Vous utiliserez deux membres de cette classe :

- une propriété **AllKeys** qui permet d'obtenir un tableau des clés
- une méthode **getValues(index)** qui permet d'obtenir les valeurs stockées à la position d'une clé.

Voir la référence MSDN à l'index **NameValueCollection**

Lorsque l'objet **HttpContext** n'est pas transmis en argument d'une méthode, vous disposez d'une propriété statique **Current** qui vous permet d'accéder à la requête en cours.

Voir les chapitres 2 et 3 du support de cours Asp Net Objets intégrés

Dans un bloc délimité par `<% %>` à l'emplacement où vous souhaitez écrire (dans le corps du document), déclarez les variables dont vous aurez besoin par la suite et affectez à celles-ci la référence des objets intrinsèques dont vous aurez besoin.

```
<%  
HttpContext requeteCourante;  
    HttpRequest requete;  
    HttpResponse reponse;  
    requeteCourante = HttpContext.Current;  
    requete = requeteCourante.Request;  
    reponse = requeteCourante.Response;  
    NameValueCollection clesValeurs = requete.QueryString;
```

...

```
%>
```

Récupérez l'ensemble des clés.

Mettre en place une structure itérative vous permettant d'extraire la ou les valeurs de chaque clé.

Ecrire dans le flux HTML en plaçant les valeurs extraites de l'objet QueryString dans un tableau.

Vous trouverez page suivante le résultat attendu.

Saisie des informations vous concernant

Sexe : ☒ Homme ☐ Femme

Nom :

Prénom :

Adresse :

..

Code Postal Ville:


Pays :

Adresse mail :

Tél. Format Inter :

Vos centres d'intérêts ☐ Bricolage ☐ Jardinage ☐ Lecture ☐ Voyages

Résultat obtenu à l'exécution de la page A002.aspx

Propriété	Valeur
rdSexe	H
txtNom	Bost
txtPrenom	vincent
txtAdresse_P1	Le bois colombes
txtAdresse_P2	
txtCodePostal	19270
txtVille	Sainte féréole
selPays	FR
txtMail	vincent.bost@gmail.com
txtTelephone	 +33555927723
chkInterets	JD ,LC ,VG

Extrait du code de la page générée :

```
<table
class='tbFormulaire'><thead><tr><td>Propriété</td><td>Valeur</td></tr></thead><tr><td>rdSexe</td><td>H</td></tr>
...
<tr><td>txtMail</td><td>vincent.bost@gmail.com</td></tr><tr><td>chkInterets</td><td>JD ,LC ,VG</td></tr></table>
```

1.3 Modification pour prendre en compte la méthode POST

Objectif : traitez les valeurs transmises par POST au lieu de GET.

Pour cela vous devez modifier la méthode du formulaire HTML puis vous pouvez envisager de modifier légèrement le code de votre page aspx pour prendre en compte les deux méthodes.

Vous pourriez par exemple tester la commande http utilisée :

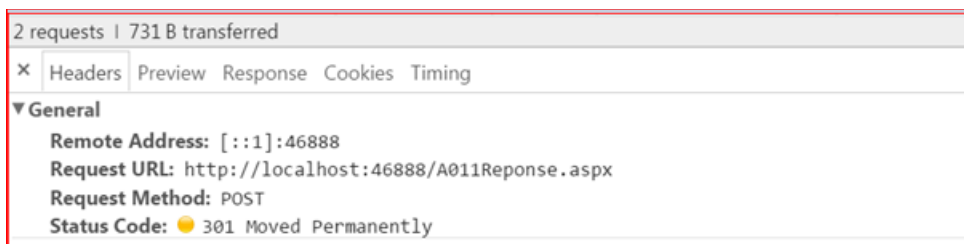
```
switch (this.Request.RequestType)
{
    case "GET":

        break;
    case "POST":

        break;
}
```

De manière analogue aux points traités précédemment, affichez les propriétés et valeurs de la collection Params de l'objet Request. Conclusion ?

Dans l'architecture .Net, la page qui traite les données est par défaut celle qui les a transmis (referrer). Le routage des demandes prévoit d'ailleurs, par défaut, une redirection en GET lorsque cette règle n'est pas respectée.



Le code 301 indique que la demande a été dirigée, le serveur répondant dans les entêtes http que la ressource a été déplacée. Une nouvelle requête Get vers la ressource est alors automatiquement effectuée mais les valeurs passées en paramètres dans le Post sont perdues.

Pour modifier ce comportement par défaut, modifier le routage des requêtes (fichier Route.config dans le répertoire App_start), AutoRedirectMode Off.

```
public static class RouteConfig
{
    1 référence
    public static void RegisterRoutes(RouteCollection routes)
    {
        var settings = new FriendlyUrlSettings();
        settings.AutoRedirectMode = RedirectMode.Off;
        routes.EnableFriendlyUrls(settings);
    }
}
```