Module 2 Développer pages web en lien avec base de données

Programmation Server Side	Séance	S03	Activité	A-003

Objectifs de cette séance :

- Etre capable de programmer des formulaires Web.

Vous allez découvrir au cours de cette séance les différents contrôles à votre disposition :

- Les contrôles serveur HTML
- Les contrôles serveur

Vous découvrirez les mécanismes de gestion d'affichage et de transmissions de données via un formulaire unique.

Sommaire de l'activité proposée :

1. Travailler avec des contrôles serveur HTML

Les contrôles HTML peuvent être remplacés par leur équivalent serveur.
Ces contrôles HTML Serveur sont alors accessibles via un champ à partir de votre page aspx.
Les attributs de l'élément HTML seront accessibles aussi bien à partir du serveur que du client.

Pour passer d'un contrôle HTML à un contrôle serveur HTML vous devez :

- Indiquez l'attribut HTML runat='server'
- Définir l'attribut id si ce n'est pas déjà fait

Pour pouvoir être accessible, les champs qui contiennent la référence aux contrôles doivent avoir pour accessibilité minimale « Protected ».
Déclarez un champ de visibilité protégé de type adéquat et de nom identique à l'id si ce mécanisme n'a pas été réalisé automatiquement par le concepteur graphique.

1.1 Modifier les contrôles HTML en contrôles serveur HTML

1. Ajoutez un nouvel élément de type Web Form
2. Sélectionnez les contrôles de la page HTML A002.html fourni en source de l'atelier et copiez-les au sein de la balise intégrée au formulaire. Modifiez les contrôles HTML de type INPUT TEXT et INPUT BUTTON en précisant l'attribut runat='server'
3. Vérifiez que votre page se compile bien et que son rendu visuel est correct.
   Affichez le code source généré.
   Voyez-vous une différence significative avec celui généré de la page A003.html ?
   Ouvrez le fichier portant l'extension .designer.cs
   Vous devriez trouver ces lignes :

   ```
   protected global::System.Web.UI.HtmlControls.HtmlInputText txtNom;
   protected global::System.Web.UI.HtmlControls.HtmlInputText txtPrenom;
   protected global::System.Web.UI.HtmlControls.HtmlInputButton btnEnvoyer;
   ```

   Vous pourrez maintenant accéder à vos contrôles HTML Server et leurs propriétés en code Behind ainsi :

   ```
   this.txtNom.Value = "Bost";
   ```

   Cette approche vous rappelle celle mise en œuvre dans les formulaires Windows.

4. Attribuez à la balise form l'attribut runat='server'. Conservez l'attribut id et supprimer les attributs method et action.

   Il n'est pas utile de spécifier une valeur pour la balise action : par défaut, le formulaire traitant les données est celui qui les poste. Ainsi, nous aurons :

   - Un premier traitement d'affichage du formulaire dans son état initial
   - Un deuxième traitement qui consistera à prendre en charge les valeurs postées.

   La méthode Post est toujours préférée à la méthode Get pour le traitement des données transmises via un formulaire. Elle permet de transmettre de plus gros volumes d'information et ne se limite pas aux contenus texte : vous pouvez ainsi transmettre des images, des documents ….

   Modifiez la directive de page pour tracer les actions exécutées. Attribut **Trace**= ''true''

   Soumettez les données de votre formulaire pour test.

   Vous devriez constatez que la page qui traite les données est la même que celle qui les transmet. Cette règle de fonctionnement par défaut peut être modifiée.

1.2 Le mécanisme de PostBack

L'objectif est ici de différencier l'affichage initial de la page de celui intervenant après le postage des données en testant la propriété de la page IsPostBack.

Remarque : Cette propriété n'est correctement alimentée que si le formulaire s'exécute côté serveur. L'attribut runat doit donc avoir la propriété 'server'.

Pour vous assurer d'une bonne compréhension de ces mécanismes et vous permettre de vous approprier les techniques de manipulation des attributs des balises vous pouvez modifier la valeur d'une boite de texte en fonction de cette propriété.

1.3 Les événements côté serveur d'un contrôle HTML Serveur.

Vous allez ici mettre en place un gestionnaire d'événement côté serveur.
Il s'agit d'un événement sur changement de la valeur d'une boite de texte.

Créer un gestionnaire (handler) d'événement de type change côté serveur qui sera exécuté sur modification du nom.
Le contrôle `HtmlInputText` dispose à cet effet d'un événement serveur qui est déclenché sur le changement de la valeur de l'attribut Text.

Il est possible d'associer le gestionnaire à l'événement **via les attributs des balises HTML** :

```
onserverchange="txtPrenom_Change"
```

Ou de manière plus classique en .net via le code C# behind

```
txtPrenom.ServerChange += new EventHandler(txtPrenom_ServerChange);
```

L'événement de modification du texte d'un contrôle de type InputText ne propose pas le rechargement de la page (autopostback). Il vous faut explicitement demander celui-ci en postant les données.

Rappelez-vous que lorsque la propriété `AutoEventWireUp` de la directive de page vaut « `true` », vous n'avez pas à associer les gestionnaires d'événement aux **événements de page**. Dans ce contexte, les handlers d'événements de page sont contraints de respecter un nom prédéfini de type page_Evenement.

Pour les gestionnaires d'événements associés aux événements des contrôles et non à la page,
Vous pouvez donc choisir un tout autre nom que `txtPrenom_ServerChange` pour le gestionnaire d'événement.

Vérifiez le bon fonctionnement de votre code en modifiant l'aspect du contrôle.

Modifiez un attribut d'apparence de la boite de texte lors de l'interception de l'événement côté serveur.