Zoom sur la création de contrôles de validation personnalisés

Ici, nous allons apprendre à créer des contrôles de validation personnalisés en respectant le patron de conception mis en œuvre dans l'architecture .Net.

Il n'existe pas de contrôle de validation qui permet de s'assurer qu'un des éléments d'une liste ait été sélectionné.

Ainsi, si vous associez la propriété ControlToValidate à un contrôle de type CheckBoxList vous obtenez une erreur :

```
<asp:CheckBoxList ID="CentreInterets" ClientIDMode="Static"
     runat="server" RepeatDirection="Horizontal"
    ValidationGroup="Interets">
    <asp:ListItem>Bricolage</asp:ListItem>
    <asp:ListItem>Lecture</asp:ListItem>
    <asp:ListItem>Voyages</asp:ListItem>
</asp:CheckBoxList>
<asp:CustomValidator ID="CheckCI" runat="server"
    ControlToValidate="CentreInterets"
    ClientValidationFunction="IsSelectionValide"
    ErrorMessage="Vous devez sélectionner au moins un centre d'intérêt"
    OnServerValidate="IsSelectionValideServer">
</asp:CustomValidator>
```

```
Le contrôle 'CentreInterets' référencé par la propriété ControlToValidate de 'CheckCI' ne peut pas être validé.
 à System.Web.UI.WebControls.BaseValidator.CheckControlValidationProperty(String name, String propertyName)
 à System.Web.UI.WebControls.CustomValidator.ControlPropertiesValid()
 à System.Web.UI.WebControls.BaseValidator.OnPreRender(EventArgs e)
```

Sommes-nous condamnés à vérifier cette règle en mettant en œuvre une fonction propre qui prendrait en charge l'affichage du message d'erreur et l'annulation de la soumission des données en cas d'erreur ?

Non.

Nous pouvons créer notre propre mécanisme de validation en respectant le pattern des contrôles de validation.

Ce document se propose de compléter l'introduction faite dans le chapitre 6.3 de la définition des contrôles personnalisés.

Côté client

La fonction côté client expose deux arguments, source et args.

Source représente les attributs de l'objet de validation. Nous y trouvons donc les attributs :

- ControlToValidate
- ClientValidationFunction
- ErrorMessage
- ....

L'argument args expose la valeur du contrôle à vérifier, obtenu grâce à l'attribut ControlToValidate et l'attribut IsValid qui permettra de définir la valeur du contrôle vérifié comme valide ou non.

Dans le cas de la vérification d'une liste, nous n'avons pas la possibilité de récupérer l'identifiant du contrôle à valider non définie (source.ControlToValidate) et ne disposons pas de la valeur.

Version 1

Nous pouvons toutefois implémenter un mécanisme qui nous permettra de dépasser les limtes du contrôle de validation personnalisé de base.

Définissons tout d'abord les propriétés du contrôle de validation personnalisé sans définir l'identifiant du contrôle à valider :

```
<asp:CustomValidator ID="CheckCI" runat="server"
    ClientValidationFunction="IsSelectionValide"
    ErrorMessage="Vous devez sélectionner au moins une valeur"
    OnServerValidate="IsSelectionValideServer">
</asp:CustomValidator>
```

Et définissons la fonction client en charge de la vérification.

```
function IsSelectionValide(source, args) {
    var checkBoxList = document.getElementById("CentreInterets");
    var checkboxes = checkBoxList.getElementsByTagName("input");
    args.IsValid = false;
    for (var i = 0; i < checkboxes.length; i++) {
        if (checkboxes[i].checked) {
            args.IsValid = true;
            break;
        }
    }
}
```

A noter : Nous récupérons la table des éléments de type input checkbox par le biais de l'identifiant du contrôle. Attention à bin préciser la génération d'un identifiant client statique :

```
<asp:CheckBoxList ID="CentreInterets" ClientIDMode="Static"
```

Extrait du HTML généré :

```
<span id="Label5">
    Choisissez un centre d'intérêt :
</span>
<table id="CentreInterets">
    <tr>
        <td><input id="CentreInterets_0" type="checkbox" name="CentreInterets$CentreInterets_0"
        <td><input id="CentreInterets_1" type="checkbox" name="CentreInterets$CentreInterets_1"
        <td><input id="CentreInterets_2" type="checkbox" name="CentreInterets$CentreInterets_2"
    </tr>
</table>
```

Cette fonction est opérationnelle mais elle a la faiblesse de nécessiter une fonction par liste de cases à cocher ou bontons radio.

Version 2

Nous pouvons pour suppléer à cette faiblesse nous appuyer sur un mécanisme redoutable de modification dynamique de scripts à partir du serveur :

Nous aurions besoin de connaître la propriété Name. Nous allons combiner la souplesse de JavaScript et les fonctionnalités avancées d'ASP.Net.

La classe Page propose un objet ClientScript qui représente un ScriptManager, à savoir un objet qui vous permet de créer, amender, gérer les scripts côté client.

Cette classe vous permet aussi de créer des fonctions de rappel du client similaires à ce qui est utilisé dans Ajax.

C'est un objet très puissant mais aussi assez complexe…..Ici, je vais utiliser une méthode qui va me permettre de fournir un attribut supplémentaire à mon objet de validation : le nom du groupe de boutons à valider.

Script Manager

Méthode pour ajouter un attribut

Id du contrôle de validation

Id du groupe

Valeur de l'attribut

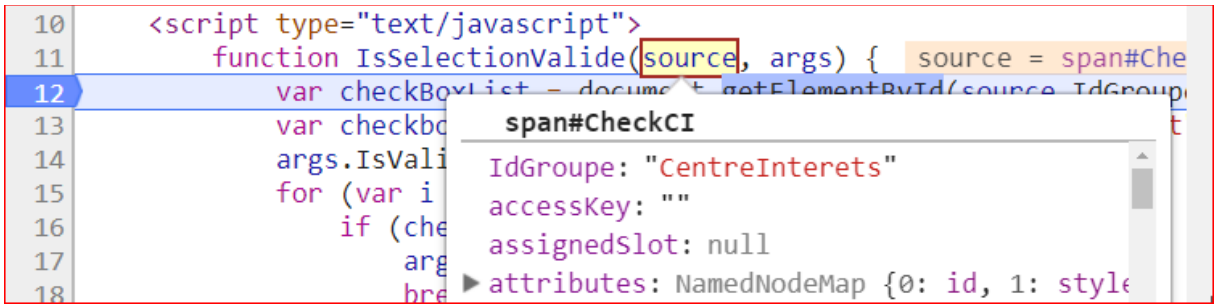
```
Page.ClientScript.RegisterExpandoAttribute(CheckCI.ClientID,
    "IdGroupe", CentreInterets.ID);
```

Cette méthode est invoquée lors de la création de la page dans le gestionnaire Page_Load.

Modifions la fonction de validation côté client en conséquence :

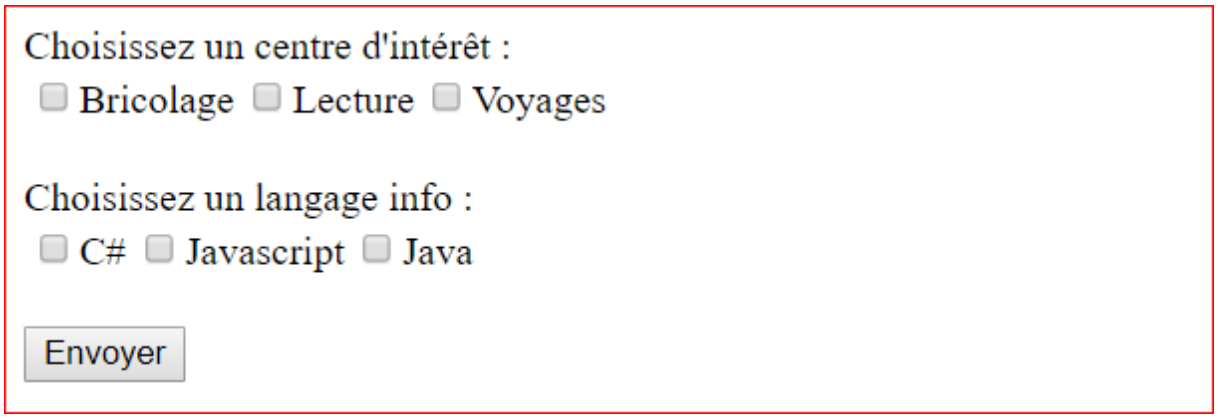
```
<script type="text/javascript">
    function IsSelectionValide(source, args) {
        var checkBoxList = document.getElementById(source.IdGroupe);
        var checkboxes = checkBoxList.getElementsByTagName("input");
        args.IsValid = false;
        for (var i = 0; i < checkboxes.length; i++) {
            if (checkboxes[i].checked) {
                args.IsValid = true;
                break;
            }
        }
    }
```

En mode Debug côté client, nous voyons un nouvel attribut IdGroupe avec comme valeur l'identifiant du groupe de contrôle.



Amendons notre page pour nous assurer que notre processus de validation fonctionne avec plusieurs listes et une seule méthode de validation (Objectif !)

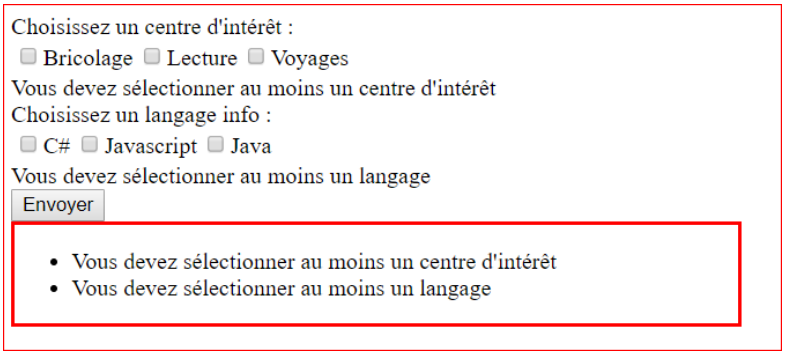
.



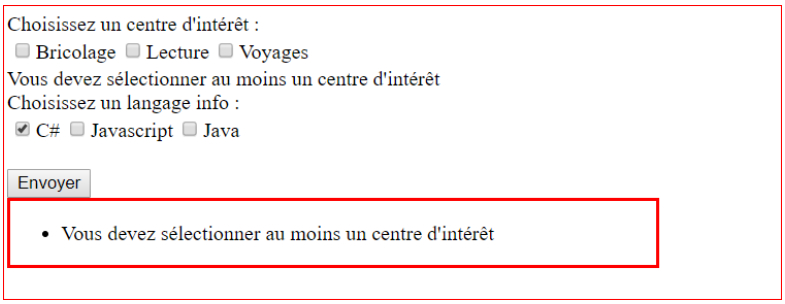
Ajoutons un attribut au groupe des langages.

```
protected void Page_Load(object sender, EventArgs e)
{
    Page.ClientScript.RegisterExpandoAttribute(CheckCI.ClientID,
        "IdGroupe", CentreInterets.ID);
    Page.ClientScript.RegisterExpandoAttribute(CheckLangages.ClientID,
       "IdGroupe", Langages.ID);
}
```

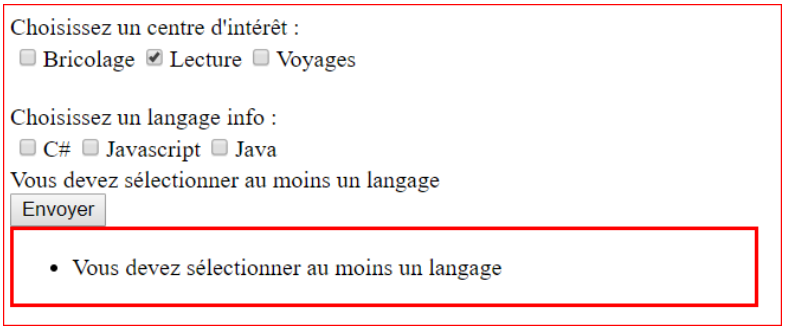
Test 1 : Aucun élément sélectionné : OK



Test 2 : Aucun centre d'intérêt sélectionné : OK



Test 3 : Aucun langage sélectionné : OK



Côté Serveur

Nous allons là encore rencontrer des difficultés similaires à celles identifiées côté client. Nous n'avons pas pu définir la propriété du contrôle **ControlToValidate**

L'objet **source** passé en argument de la méthode de validation est de type **CustomValidator**. Mais sa propriété **ControlToValidate** ne pouvant être exploitée (non définie), je suis obligé de désigner le contrôle à valider.

Première version :

```
protected void IsSelectionValideServer(object source,
    ServerValidateEventArgs args)
{
    args.IsValid = false;
    foreach (ListItem item in CentreInterets.Items)
    {
        if (item.Selected)
        {
            args.IsValid = true;
            break;
        }
    }
}
```

Comme pour la validation côté client, cette fonction est opérationnelle mais spécialisée pour les centres d'intérêts. Si nous devons valider plusieurs groupes de contrôles, il nous faut alors créer une fonction par groupe.

Plusieurs solutions peuvent être envisagées pour implémenter une fonction de validation générique côté serveur:

- Créer un contrôle de validation dérivé et ajouter une propriété à ce dernier qui nous permettrait de spécifier le groupe à valider.
- Ajouter un attribut à la balise, attribut dont la vaaleur déterlinerait le groupe à valider

La première solution serait certainement la plus pertinente mais elle demande un effort de programmation supplémentaire. Nous verrons la programmation des contrôles utltérieurement.

Nous allons ici utiliser la possibilité offerte d'ajouter des attributs supplémentaires aux balises à nos propres fins. Nous pouvons recourir à cette option dans bien des contextes pour enrichir les fonctionnalités des contrôles web.

Modification des contrôles de validation :

Ajout d'un attribut ControleAssocie

```
<asp:CustomValidator ID="CheckCI" runat="server"
    ControleAssocie="CentreInterets"
    ClientValidationFunction="IsSelectionValide"
    ErrorMessage="Vous devez sélectionner au moins un centre d'intérêt"
    OnServerValidate="IsSelectionValideServer">
</asp:CustomValidator>
```

Nous réalisons la même opération avec la liste des langages.
Puis modifions la méthode de validation côté serveur :

```
protected void IsSelectionValideServer(object source, ServerValidateEventArgs args)
{
    CustomValidator cV = source as CustomValidator;
    // Récupération de l'attribut ajouté
    string controleID = cV.Attributes["ControleAssocie"];
    // Récupération du contrôle
    CheckBoxList listeCB = this.FindControl(controleID) as CheckBoxList;
    args.IsValid = false;
    foreach (ListItem item in listeCB.Items)
    {
        if (item.Selected)
        {
            args.IsValid = true;
            break;
        }
    }
}
```

A noter : La recherche du contrôle se fait à l'aide de la méthode **FindControl**.

Pour étendre notre modèle à des groupes de boutons, nous pouvons utiliser la classe **ListControl** au lieu de **CheckBoxList**. **RadioButtonList** et **CheckBoxList** héritent de cette même classe.

```
// Récupération du contrôle
ListControl listeCB = this.FindControl(controleID) as ListControl;
```

Vérifions le bon fonctionnement en appelant la méthode IsValid de la page.

```
protected void btnEnvoyer_Click(object sender, EventArgs e)
{
   // Page.Validate("Interets");
    if (this.Page.IsValid)
    {

    }
}
```