Les contrôles de validation

- RequiredFieldValidator
- CompareValidator: comparer la valeur saisie avec une valeur constante, la valeur de la propriété d'un autre contrôle ou bien une extraite à partir d'une base de données.
- RangeValidator
- RegularExpressionValidator
- CustomValidator

RequiredFieldValidator

```
<asp:RequiredFieldValidator
  ID="RequiredFieldValidator1" runat="server"
  ControlToValidate="TextBox1" ErrorMessage="La
  saisie du nom est obligatoire">
</asp:RequiredFieldValidator>
```

- Autres propriétés:
  - InitialValue: valeur initial du contrôle à valider

RangeValidator

```
<asp:RangeValidator
ID="rv1"
runat="server"
ControlToValidate="TextBox2"
ErrorMessage="La valeur doit être comprise ente 1
   et 1000"
MaximumValue="1000"
MinimumValue="1" Type="Integer">
</asp:RangeValidator>
```

- Type : Integer, String, Date, Double, Currency

CompareValidator

- Ce contrôle peut être utilisé soit pour valider le type de données du champ associé soit pour comparer sa valeur par rapport à une valeur de référence (la valeur de référence peut être définie par une constante ou par la valeur d'un autre champ du formulaire)
- Propriétés
  - Operator: DataTypeCheck,Equal, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual et NotEqualType
- Type: String, Integer, Double, Date, and Currency.
  - Exemple 2:

```
void Page_Load()
{
    cmpDate.ValueToCompare = DateTime.Now.ToString("d");
}
```

```
<asp:CompareValidator
    id="cmpDate"
    Text="La date doit être plus récente que la date actuelle"
    ControlToValidate="txtDate"
    Type="Date"
    Operator="GreaterThan"
    Runat="server" />
```

  - Exemple3: Comparaison entre les valeurs de deux champs

```
<asp:CompareValidator
    id="cmpDate"
    Text="(Plage de date non valide)"
    ControlToValidate="txtDateFin"
    ControlToCompare="txtDateDebut"
    Type="Date"
    Operator="GreaterThan"
    Runat="server" />
```

RegularExpressionValidator

- Exemple

```
<asp:RegularExpressionValidator
        id="regEmail"
        ControlToValidate="txtEmail"
        Text="(L'adresse email n'est pas valide)"
        ValidationExpression =" \w+([-+.']\w+)*@\w+([-
    .]\w+)*\.\w+([-.]\w+)* "
        Runat="server" />
```

CustomValidator

- Propriétés
  - **ClientValidationFunction: validation côté client**
  - **ServerValidate: validation côté serveur**
- **Il n'est pas obligatoire d'associer un contrôle CustomValidator à un champ.**
- Exemple

```
<script runat="server">
// Validation côté serveur
    void valComment_ServerValidate(Object source,
    ServerValidateEventArgs args)
    { // args est une instance de la classe ServerValidateEventArgs qui
    possède les propriétés
     - value:  valeur du champ en cours de validation
     - IsValid:
     - ValidateEmptyText: détermine si le champ sera validé même s'il est
    vide ou non
        if (args.Value.Length > 10)
            args.IsValid = false;
        else
            args.IsValid = true;
    }
</script>
```

CustomValidator

```
<asp:TextBox
    id="txtComment"
    TextMode="MultiLine"
    Columns="30"
    Rows="5"
    Runat="server" />
<asp:CustomValidator
    id="valComment"
    ControlToValidate="txtComment"
    Text="(Un commetaire doit comporter au moins 30 caractères)"
    OnServerValidate="valComment_ServerValidate"
    Runat="server" />
```

- On peut aussi définir une validation côté client.

```
<script type="text/javascript">

    function valComments_ClientValidate(source, args)
    {
        if (args.Value.length > 10)
            args.IsValid = false;
        else
            args.IsValid = true;
    }

</script>
```

Et

```
…
ClientValidationFunction="valComments_ClientValidate"
…
```

ValidationSummary

- Affiche les messages d'erreurs définis par la propriété ErrorMessage de tous les contrôles de validation.
- Propriétés
  - DisplayMode: définit le format d'affichage des messages d'erreur.
  - HeaderText
  - ShowMessageBox: Pour afficher les messages dans une boîte de dialogue.
  - ShowSummary: Afficher/Masquer l'affichage des messages d'erreur.

Remarques

- La validation est assurée au niveau du client et aussi côté serveur
- Propriétés des contrôles de validation
  - EnableClientScript: (par défaut = true) si elle est égale à false alors la validation javascript est désactivée pour le contrôle associé.
  - Chaque contrôle de validation possède la propriété IsValid qui est évaluée à true s'il n'y a pas d'erreur de validation.
  - Display: définit comment le contrôle sera généré en html:
    - Static: La saisie du champ est obligatoire
    - Dynamic: La saisie du champ est obligatoire
    - None: les message d'erreur peut être affiché uniquement dans un contrôle ValidationSummary.
    - Text: Message d'erreur à afficher (peut contenir de l'HTML).
    - SetFocusOnError
    - ValidationGroup: permet de créer des groupes de contrôles de validation, utile dans le cas d'une page contenant plusieurs formulaires, dans ce cas les contrôles de validation et le bouton d'envoi de chaque formulaire doivent avoir la même valeur dans la propriété ValidationGroup:
- La propriété Page.IsValid est égale à true lorsque tous les contrôles de validation ont la propriété IsValid égale à true.

Créer un contrôle de validation personnalisé: la classe BaseValidator

- Pour créer un contrôle de validation personnalisé:
  - Créer une classe qui dérive de la classe de base abstraite BaseValidator
  - Définir la méthode EvaluateIsValid qui doit retourner true si le champ associé au contrôle est valide.
- La classe BaseValidator contient d'autres méthodes qu'on peut surcharger ou bien utiliser :
  - GetControlValidationValue(): retourne la valeur du champ en cours de validation.
- Exemple soit le contrôle TValidation qui permet de définir la taille maximale d'un champ.

```
namespace controles_validation
{ public class TValidator :
    System.Web.UI.WebControls.BaseValidator  {
        private int _tMax = 0;
        public int TMax
        {
            get
```

Créer un contrôle de validation personnalisé: la classe BaseValidator

```
        {  return _tMax; }
        set
        { _tMax = value;}
    }
    protected override bool EvaluateIsValid()
    { string value =
this.GetControlValidationValue(this.ControlToValidate);
        if (value.Length > _tMax)
            return false;
        else
            return true; }}}
```

- La classe doit être ajoutée dans le dossier App_Code (toute classe contenue dans ce dossier est compilée automatiquement par le framework .Net)
- La classe peut enregistrée ou bien dans la directive @ register de la page ou bien dans la section <pages> du fichier web.config