**Validation des données**

La validation des données est en général la chose la plus importante dans un site web

Il faut travailler côté client et côté serveur pour prévenir au plus tôt l'utilisateur d'une erreur éventuelle

Il existe dans le Framework des contrôles dédiés à la validation. C'est-à-dire qu’il existe des contrôles permettant, par exemple, la vérification des champs d’un formulaire : si le champ est vide, si les deux mots de passe correspondent, si le champ correspond aux expressions régulières, etc.

## RequiredFieldValidator

Le plus fréquemment utilisé car il est le ***seul*** qui peut s'assurer qu'un champ n'est pas vide

Tous les autres contrôles de validation acceptent un champ vide donc, associer ce contrôle de validation aux autres contrôles permet cette vérification essentielle

|  |  |
| --- | --- |
| **Nom de la propriété** | **Utilisation** |
| ControlToValidate | doit contenir le nom du contrôle à valider |
| ErrorMessage | message à afficher en cas d'erreur dans le contrôle ValidationSummary |
| InitialValue | contient une valeur qui invalide le contrôle si celui-ci est égal à cette valeur précise |
| Text | texte affiché en cas de non validation |

Exemple

si on clic sur le bouton sans avoir mis de valeur dans la TextBox, le message d’erreur sera affiché, comme le montre l’image ci-contre

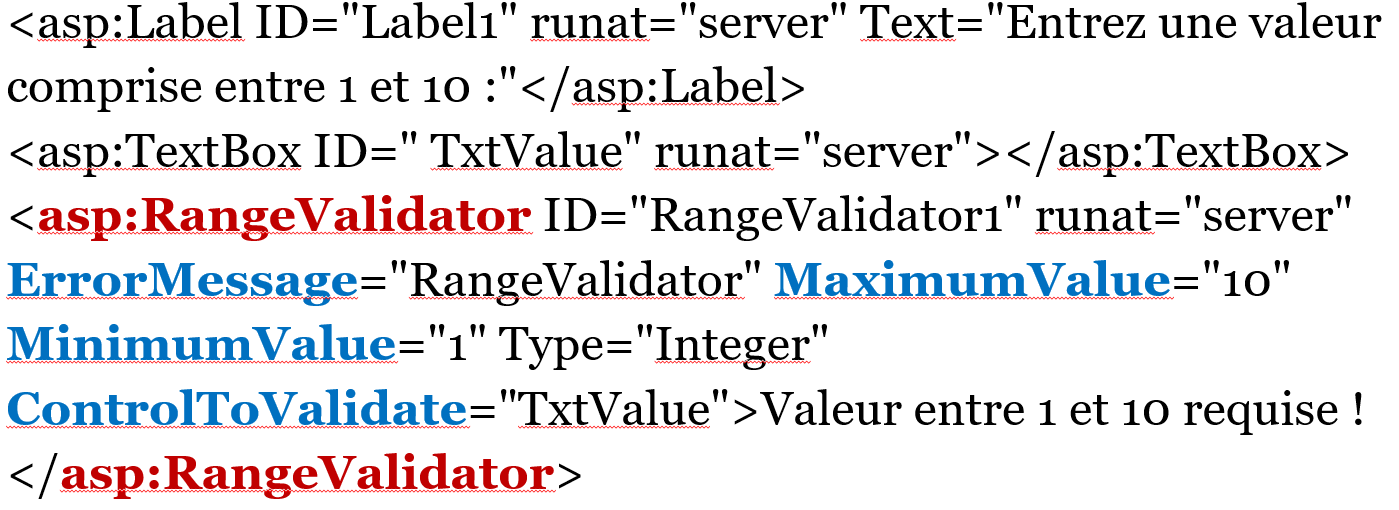


## RangeValidator

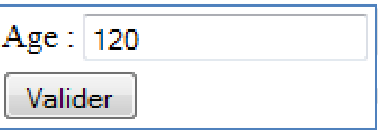
* + Comme son nom l'indique, il sera utilisé pour valider l'encodage entre des bornes données (par exemple encoder un nombre entre 1 et 10)
  + Les propriétés sont pratiquement identiques à celles du contrôle précédent

|  |  |
| --- | --- |
| **Nom de la propriété** | **Utilisation** |
| ControlToValidate | doit contenir le nom du contrôle à valider |
| ErrorMessage | message à afficher en cas d'erreur dans le contrôle *ValidationSummary* |
| MinimumValue | valeur minimale de la plage de données |
| MaximumValue | valeur maximale de la plage de données |
| Text | texte affiché en cas de non validation |

* Exemple de validation entre 1 et 10



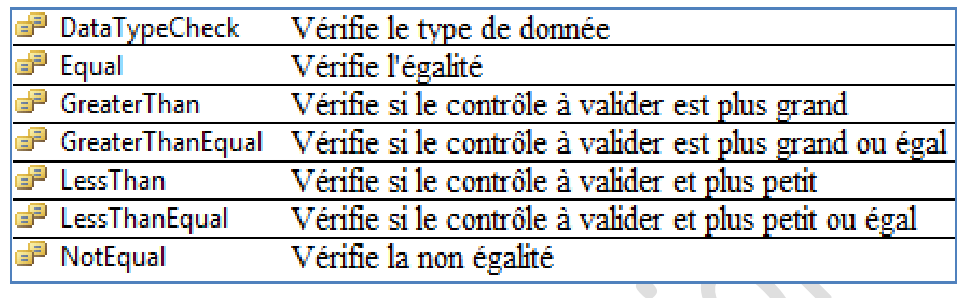
Exemple



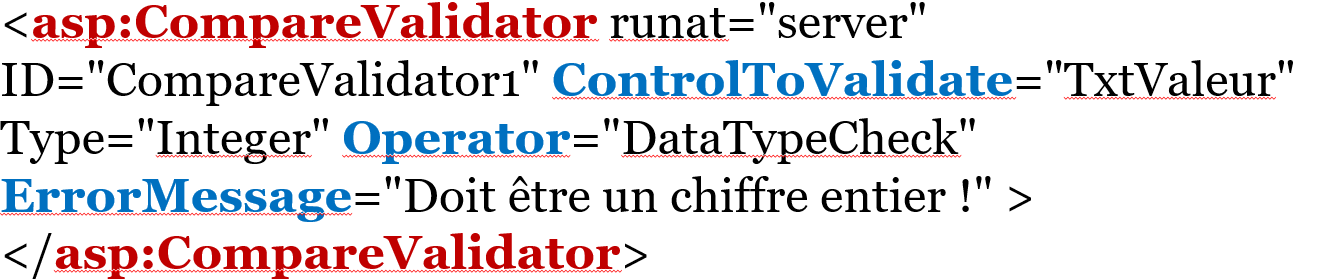
## CompareValidator

* + Il utilise un opérateur pour comparer les valeurs présentes et valider leur concordance
  + La situation la plus courante d'utilisation est, bien entendu, lors d'une deuxième saisie d'un mot de passe

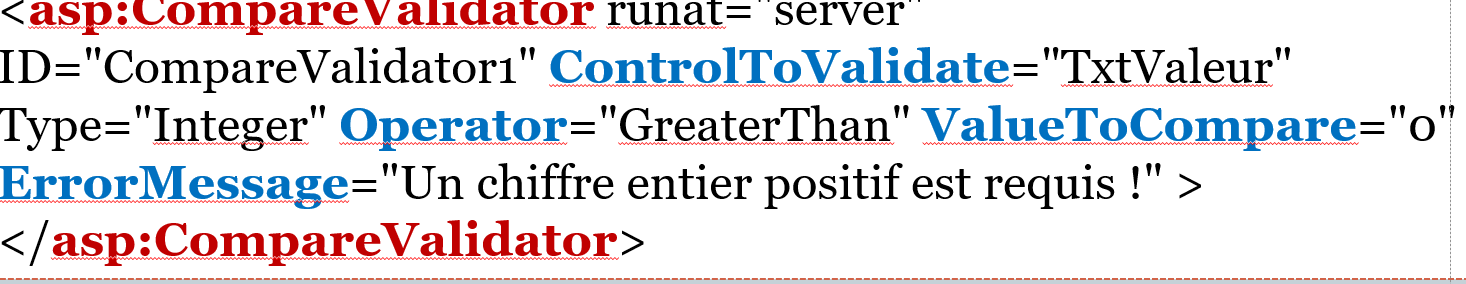
Comme on l’a dit on peut comparer avec un autre contrôle. Mais on peut aussi comparer avec une valeur que l’on rentrera nous-même. La valeur DataTypeCheck permet de comparer le type des données ce qui n’est pas le cas par défaut. Il faut donc spécifier le type à l’aide de la propriété Type du contrôle de validation : Currency (valeur monétaire), Date, Double, Integer et String. Pour les autres opérateurs, il faut obligatoirement une valeur de comparaison contenu soit dans un contrôle spécifié avec ControlToCompare, soit avec une valeur que l’on indique dans ValueToCompare.



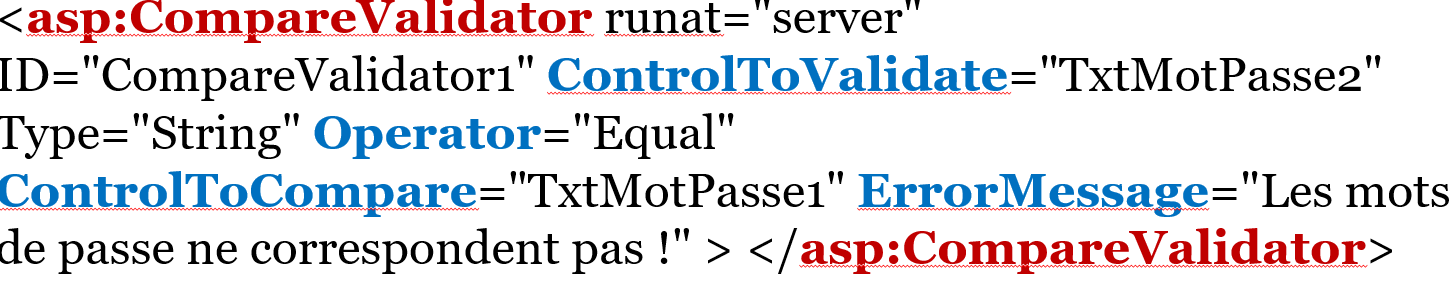
* + - Comparaison à un type



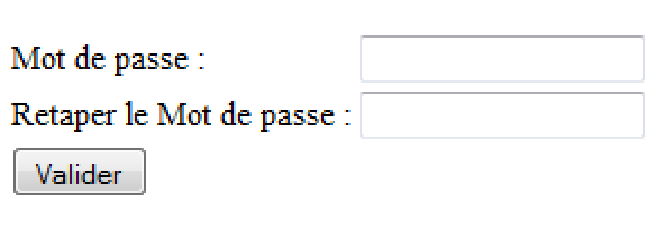
* + - Comparaison à une valeur



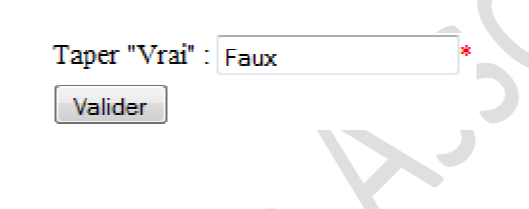
* + - Comparaison à un autre champ



Exemple 1

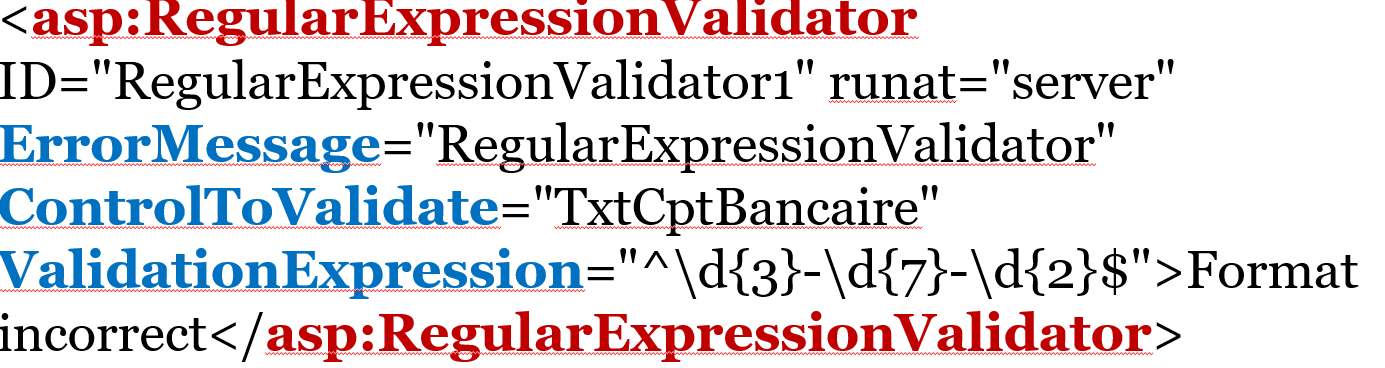


Exemple2



## RegularExpressionValidator

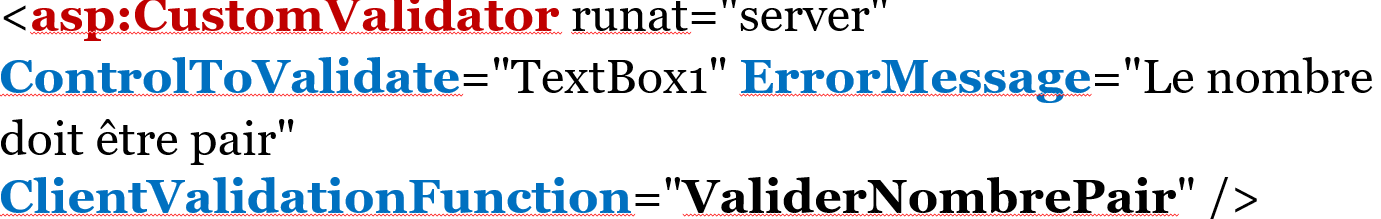
* + Ce contrôle valide un champ suivant une expression régulière
* Un exemple de validation d'un numéro de compte bancaire



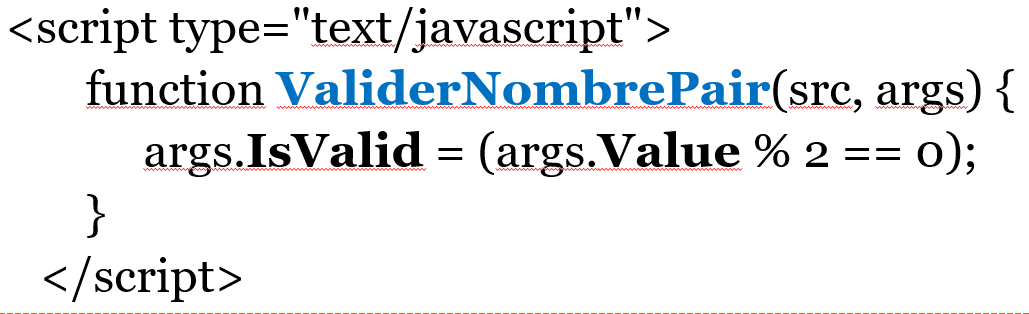
## CustomValidator

* + L'utilisateur définit lui-même une fonction pour effectuer la validation lorsque les contrôles standards ne peuvent pas assumer ce rôle
  + Dans ce cas, les propriétés sont un peu différentes :
    - Dans le cas d'une validation côté client :
      * La propriété **ClientValidationFunction** contient le nom de la fonction
      * La fonction doit être sous la forme : *Function ValidationPersonnelle (source, arguments)*
        + La **source** est l'objet *CustomValidator* côté client

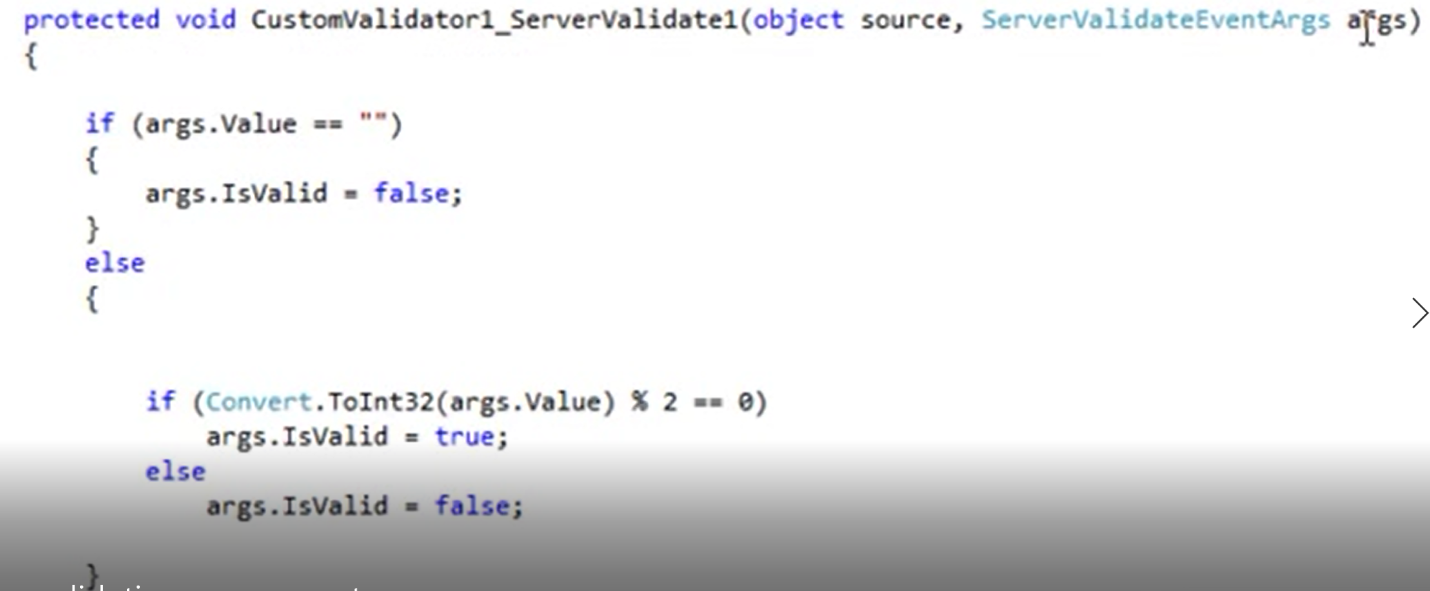
Les **arguments** est un objet comportant deux propriétés : *Value* (valeur à valider) et *IsValid* (booléen retournant le résultat de la validation )



On ajoute le code javascript suivant :



Ou bien ajouter l’évènement



## ValidationSummary

Ce contrôle n'est pas un contrôle de validation à proprement parler, il sert à afficher sous différentes formes le résultat de tous les contrôles de validation sur la page aspx si une erreur est survenue

Il est bien évident que vous pouvez l'omettre et gérer vous-même un affichage d'erreur

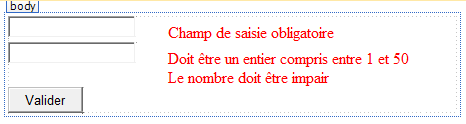
Le contrôle *ValidationSummary* s'affiche dès que la propriété *IsValid* de la page est à *False*

* + Pour afficher le résultat, vous avez les *DisplayMode* suivants à votre disposition :
    - ***List*** : simple liste
    - ***BulletList*** : liste avec puces
    - ***SingleParagraph*** : les messages d'erreur sont concaténés les uns à la suite des autres, séparés par une virgule
  + L'emplacement de l'affichage peut s'effectuer de deux manières :
    - À l'emplacement du contrôle *ValidationSummary* : mettre sa propriété *ShowSummary = True*
    - Dans une boite de dialogue : mettre sa propriété *ShowMessageBox = True* (et *ShowSummary = False*)

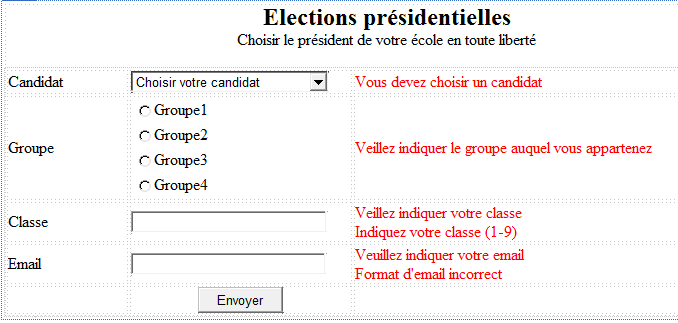
## Exercices

**Exercice 1**

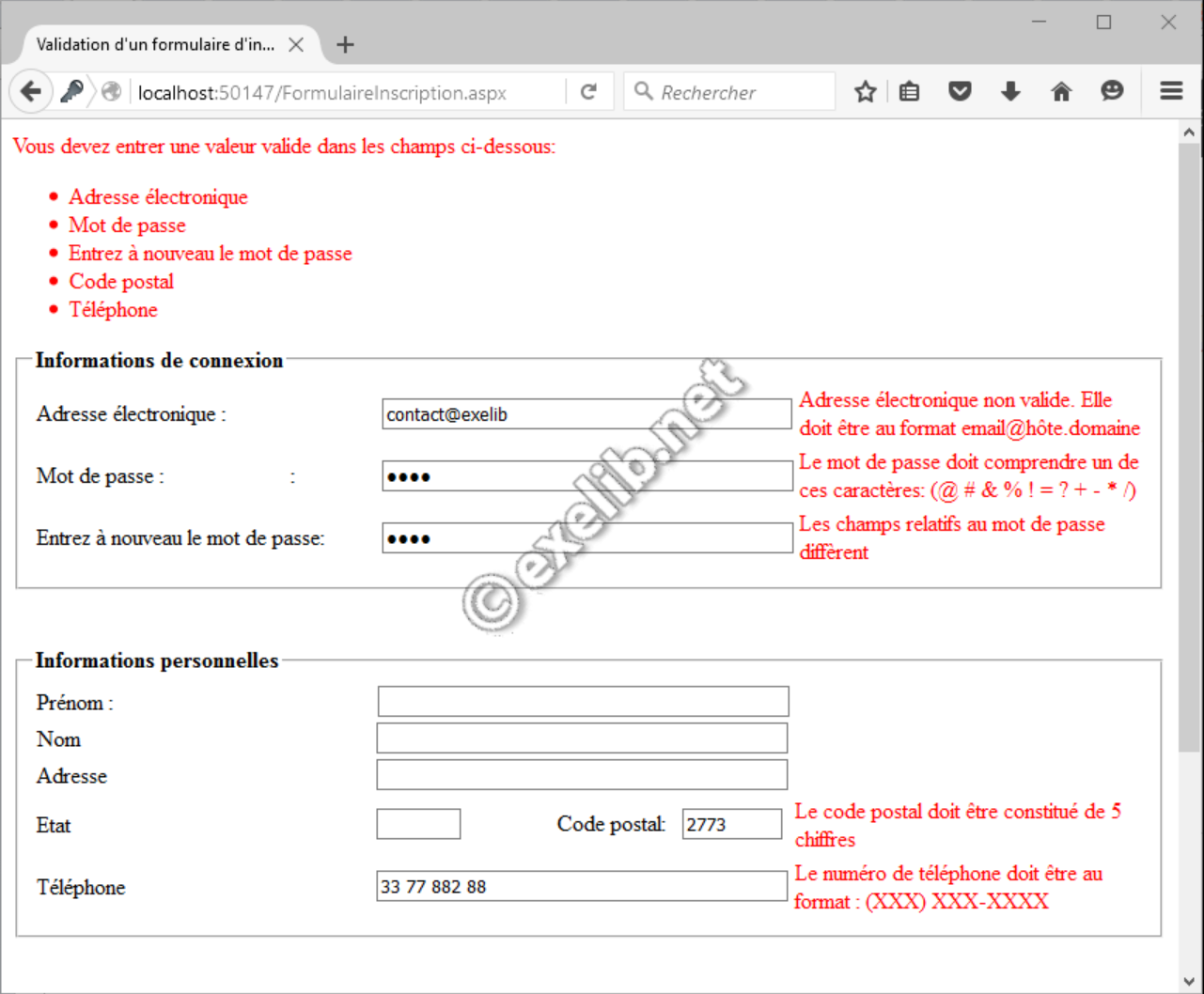
* Créer un formulaire avec deux zones de texte (TextBox1 et TextBox2)
* Ajouter un RequiredFieldValidator qui valide la TextBox1 en affichant le message « *Champ de saisie obligatoire* » en cas de manque de valeur
* Ajouter un RangeValidator qui exige une valeur entière comprise entre 1 et 50 pour la TextBox2
* Tester les validateurs des deux champs sans avoir saisi de valeur dans la TextBox1 et en insérant une valeur erronée dans la TextBox2 (<1 ou >50)
* Ajoutez un CustomValidator pour la TextBox2 qui exige une valeur impaire

****

**Exercice 2**

****

**Exercice 3**

****