

Université Libanaise

Faculté de Technologie
Génie des Réseaux Informatiques
et Télécommunications



الجامعة اللبنانية

كلية التكنولوجيا

قسم هندسة شبكات المعلوماتية والاتصالات

Architecture Client Serveur

Le Langage ASP.NET - Validation des données



Préparé par:

Dr. Youssef ROUMIEH

E-mail : youssef.roumieh@gmail.com

Reference: Web Applications Development with Microsoft .NET Framework 4.

Copyright © 2010 by Tony Northrup and Mike Snell. ISBN: 978-0-7356-2740-6

Les problèmes rencontrés et les oublis constatés dans ce support sont à signaler à youssef.roumieh@gmail.com

Introduction

- Une des étapes les plus importantes pour obtenir les données entrées par un utilisateur est de s'assurer que les données sont valides.
- La validité est déterminée par un certain nombre de critères:
 - Est-ce que l'utilisateur à entrer quelque chose?
 - Est-ce le type de données entrées est approprié (un numéro de téléphone, par exemple)?
 - Les données sont dans une fourchette requise?
- ASP.NET fournit des contrôles de validation pour vous aider à vérifier les entrées des données de formulaire Web en fonction de critères avant que les éléments de données sont acceptées.
- Cette leçon vous apprend à utiliser les contrôles de validation pour attraper des données invalides et diriger l'utilisateur à régler les problèmes qui surviennent.

Utilisation de la validation

- Les contrôles de validation vérifient la validité des données entrées dans des contrôles serveur associé sur le client avant que la page est publiée au serveur.
- Cela représente une amélioration importante aux schémas de validation précédente – la plupart des problèmes de validité peut être relevées et corrigées par l'utilisateur sans un aller-retour au serveur.
- Par conséquent, les contrôles de validation ont également fournit automatiquement la validation côté serveur. La validation côté serveur est toujours effectuée, si la validation côté client est survenue ou non.

Les contrôles de validation d'ASP.NET

- Les contrôles de validation vérifient la valeur du contrôle serveur spécifiée dans leur propriété **ControlToValidate**.

Validation control	Use to
RequiredFieldValidator	Check whether a control contains data
CompareValidator	Check whether an entered item matches an entry in another control
RangeValidator	Check whether an entered item is between two values. MinimumValue & MaximumValue - Type = Integer, String ...
RegularExpressionValidator	Check whether an entered item matches a specified format. ValidationExpression= “\d{2}-\d{4}”
CustomValidator	Check the validity of an entered item using a client-side script or a server-side code, or both
ValidationSummary	Display validation errors in a central location or display a general validation error description

Pour utiliser les contrôles de validation, suivez ces étapes:

- Dessinez un contrôle de validation sur un formulaire Web et définissez sa propriété **ControlToValidate** au contrôle que vous voulez valider.
 - Si vous utilisez le contrôle **CompareValidator**, vous devez également spécifier la propriété **ControlToCompare**.
- Définissez la propriété **ErrorMessage** du contrôle de validation au message d'erreur que vous voulez apparaître si les données du contrôle n'est pas valide.
- Définir la propriété **Text** du contrôle de validation si vous souhaitez que le contrôle de validation affiche un message autre que le message dans la propriété **ErrorMessage** quand une erreur survient.
 - Réglage de la propriété **Text** vous permet brièvement d'indiquer où l'erreur s'est produite sur le formulaire et afficher la propriété **ErrorMessage** plus longue dans un contrôle **ValidationSummary**.
- Dessinez un contrôle **ValidationSummary** sur le formulaire Web pour afficher les messages d'erreur provenant des contrôles de validation dans un seul endroit.
- Fournir un contrôle qui déclenche un événement de publication (postback).
Bien que la validation se produit du côté du client, la validation ne commence pas avant une publication est demandée.

Exemple

Simple Validation

<input type="text"/>	Required.
<input type="text" value="Test"/>	Must match Previous field
<input type="text" value="99"/>	Must be between 1 and 10.
<input type="text" value="03-665"/>	Must have the form ##-#####
<input type="button" value="OK"/> <input type="button" value="Cancel Validation"/>	

Cancel Validation

- Utilisation du contrôle ValidationSummary (propriété: HeaderText)

Simple Validation

<

Teste

<

99

<

03-826

OK

Cancel Validation

Please correct these problems:

- Required.
- Must match Previous field
- Must be between 1 and 10.
- Must have the form ###-#####


```

<form id="form1" runat="server">
    <div>
        Simple Validation<br />
        <hr />
        <br />
        <asp:TextBox ID="TextBox1" runat="server" Width="155px"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
            ControlToValidate="TextBox1" Text = "<"
            ErrorMessage="Required."></asp:RequiredFieldValidator>
        <br />
        <asp:TextBox ID="TextBox2" runat="server" Width="155px"></asp:TextBox>
        <asp:CompareValidator ID="CompareValidator1" runat="server"
            ControlToCompare="TextBox1" Text = "<" ControlToValidate="TextBox2"
            ErrorMessage="Must match Previous field"></asp:CompareValidator>
        <br />
        <asp:TextBox ID="TextBox3" runat="server" Width="155px"></asp:TextBox>
        <asp:RangeValidator ID="RangeValidator1" runat="server"
            ControlToValidate="TextBox3" Text = "<" ErrorMessage="Must be between 1 and 10."
            MaximumValue="10" MinimumValue="1" Type="Integer"></asp:RangeValidator>
        <br />
        <asp:TextBox ID="TextBox4" runat="server" Width="155px"></asp:TextBox>
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
            ControlToValidate="TextBox4" Text = "<" ErrorMessage="Must have the form ##-#####"
            ValidationExpression="\d{2}-\d{6}"></asp:RegularExpressionValidator>
        <br />
        <asp:Button ID="Button1" runat="server" Text="OK" /> &nbsp;
        <asp:Button ID="Button2" runat="server" Text="Cancel Validation"
            Width="123px" />
        <br />
        <asp:ValidationSummary ID="ValidationSummary1" runat="server"
            HeaderText="Please correct these problems:" />
    </div>
</form>

```

Simple Validation

	<
Teste	<
99	<
03-826	<
<input type="button" value="OK"/> <input type="button" value="Cancel Validation"/>	

Please correct these problems:

- Required.
- Must match Previous field
- Must be between 1 and 10.
- Must have the form ##-#####

La boîte de dialogue des erreurs de validation.

- Pour afficher les erreurs de validation en tant que boîte de dialogue, définissez la **propriété ShowMessageBox** du contrôle **ValidationSummary** à **True**.

The screenshot shows a web browser window with the address bar displaying 'localhost:2122/Validation3.aspx'. The main content area contains a form titled 'Simple Validation' with four text input fields. The first field is empty, the second contains 'Teste', the third contains '99', and the fourth contains '03-826'. Below the fields are 'OK' and 'Cancel Validation' buttons. A 'ValidationSummary' control is visible at the bottom of the form, displaying a list of error messages: 'Please correct these problems:', 'Required.', 'Must match Previous field', 'Must be between 1 and 10.', and 'Must have the form ##-#####'. A modal dialog box is open over the form, titled 'La page à l'adresse localhost:2122 indique :'. It contains the same error messages and an 'OK' button.

- TIP:** Utilisez **InitialValue** du contrôle **RequiredFieldValidator** pour ignorer les instructions incluses dans le contrôle à valider.
 - Par exemple, si le contrôle de validation est un **DropDownList** qui comprend un élément dont le texte est **Sélectionnez un élément**, saisissez **Sélectionnez un élément** dans la propriété **InitialValue** du contrôle **RequiredFieldValidator** pour faire cette sélection non valide.

Combinaison des validations: plusieurs validateurs

- Un contrôle de serveur peut avoir plusieurs validateurs.
 - Par exemple, un contrôle TextBox pour un numéro de téléphone peut être à la fois un champ obligatoire et une expression régulière, donc la zone de texte sera vérifiée par deux contrôles de validation
- Tous les contrôles de validation, sauf le contrôle RequiredFieldValidator sont considérés comme valides si elles sont vides.
 - Cela permet de fournir un message d'erreur si la commande est vide, à l'aide du contrôle RequiredFieldValidator, et un message d'erreur différent si le contrôle est hors de portée ou n'est dans le format approprié.

Annulation de validation

- Parce que la validation se produit avant que le serveur traite la page, un utilisateur peut être piégé par la validation à moins que vous fournir un moyen d'annuler la validation sans poster en arrière.
- Pour permettre à l'utilisateur d'annuler la validation, fournir un contrôle Submit HTML qui définit l'attribut `Page_ValidationActive`

```
<INPUT id="butCancel" onclick="Page_ValidationActive=false;" type="submit" value="Cancel">
```
- Cette définition annule la validation et poste la page vers le serveur.
 - Vous pouvez déterminer si l'utilisateur a annulé l'opération en vérifiant la propriété `IsValid` de l'objet `Page` dans la procédure d'événement `Page_Load`.
 - Vous devez revalider la page, car l'annulation de validation met `IsValid` à `False`.

- Le code suivant montre comment vérifier si l'utilisateur a annulé la validation:

Visual C#

```
private void Page_Load(object sender, System.EventArgs e)
{
    // Validate in case user cancelled validation.
    if (Page.IsPostBack)
    {
        Page.Validate();
        if (!Page.IsValid)
            // User cancelled validation.
            Response.Redirect("default.htm");
    }
}
```

Personnalisation de la validation

- Pour effectuer de validation des types complexes non fournie par le contrôle de validation standard, utilisez un contrôle CustomValidator et écrire du code pour effectuer la validation du côté serveur et (éventuellement) sur le côté client.
- Sur le côté du serveur, placez votre code de validation dans la procédure événementielle **ServerValidate**. Les arguments de cette procédure permettent d'accéder au contrôle à valider.
- The following event procedure validates that an entered number is prime:

```

private void vldtxtPrime_ServerValidate(object source,
    System.Web.UI.WebControls.ServerValidateEventArgs args)
{
    try
    {
        // Get value from ControlToValidate (passed as args).
        int iPrime = Int32.Parse(args.Value);
        for (int iCount = 2; iCount <= (iPrime / 2); iCount++)
        {
            // If number is evenly divisible, it's
            // not prime, return False.
            if((iPrime % iCount) == 0)
            {
                args.IsValid = false;
                return;
            }
            // Number is prime, return True.
            args.IsValid = true;
            return;
        }
    }
    catch(Exception e)
    {
        // If there was an error parsing, return False.
        args.IsValid = false;
        return;
    }
}

```

- To provide client-side validation, specify a validation script in the CustomValidator control's ClientValidationFunction property. Client-side validation is optional, and if you provide it, you should maintain similar validation code in both places. The following script provides a client-side version of the prime number validation performed on the server:

Jscript

```
<script language="jscript">
    function ClientValidate(source, arguments)
    {
        for (var iCount = 2; iCount <= arguments.Value / 2; iCount++)
        {
            // If number is evenly divisible,
            // it's not prime. Return false.
            if ((arguments.Value % iCount) == 0)
            {
                arguments.IsValid = false;
                return false;
            }
        }
        // Number is prime, return True.
        arguments.IsValid = true;
        return true;
    }
</script>
```

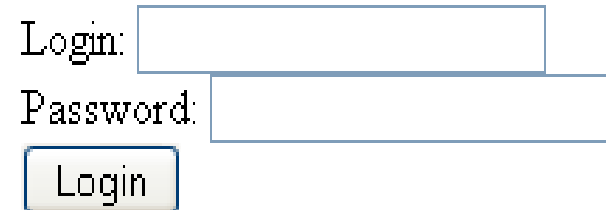

Architecture Client Serveur - TD 9 et 10

- On considère la base de données “*Bibliothèque*” suivante:
 - **Utilisateurs** (login, Nom, MotDePasse, Type),
 - **Livres** (ISBN, Titre, Catégorie, prix, auteur),
 - **LivresAchetés** (Login, ISBN, DateAchat),
 - **Categorie**(id, libellé)
- Exercice 1 : Page Login.aspx

Cette page permet de distinguer entre deux types d'utilisateurs 'normal' et 'admin' (la valeur de l'attribut Type de la table Utilisateur). Ainsi, elle permet d'identifier les utilisateurs de la base ; si le mot de passe saisi ne correspond pas au login saisi, la page Login.aspx est affichée avec le message d'erreur « Login ou Mot de passe Incorrects ». Sinon, cette page est dirigée vers la page :

- 'User.aspx' si l'utilisateur est de type normal.
- 'Admin.aspx' si l'utilisateur est de type admin.

Quelque soit le type d'utilisateur, son login est envoyé vers la page User.aspx ou Admin.aspx à partir de QueryString. En plus, s'il s'agit d'un admin, le type est envoyé par un cookie. Le nom d'utilisateur est envoyé par Session.



The screenshot shows a web form with two text input fields. The first field is labeled 'Login:' and the second is labeled 'Password:'. Below these fields is a button labeled 'Login'.

Exercice 2 : Page Admin.aspx

Sélectionner une catégorie:

Sélectionner un livre:

ISBN:
ISBN doit être entré sous le format x-xxxx-xxxx-x

Titre: *

Prix: **numeric**

Auteur:

- Cette page permet à l'administrateur de gérer la bibliothèque. Tout d'abord, il faut vérifier si le type de l'utilisateur est un '*admin*' (à partir du cookie créé dans l'exercice 1).
- L'administrateur du site peut choisir une catégorie des livres parmi la liste. Une fois qu'une catégorie est choisie, tous les livres appartenant à la catégorie choisie sont affichés dans une autre liste. Ainsi, le bouton '*Ajouter*' est affiché (voir figure, cas 1).
- Si l'administrateur choisit un livre parmi la liste des livres, alors toutes les informations concernant le livre choisi sont affichées dans les contrôles spécifiés. Dans ce cas, le bouton '*Ajouter*' disparaît et les deux boutons '*Modifier*' et '*Supprimer*' apparaissent (voir figure, cas 2).
- ISBN représente l'identifiant du Livre. L'ISBN a un format spécifique. Afin de vérifier ce dernier, deux contrôles ont été rajoutés : le premier pour s'assurer que l'ISBN a toujours une valeur et le second pour vérifier que l'ISBN est entré sous le format x-xxxx-xxxx-x où x représente un chiffre. Ainsi, nous souhaitons que le titre du livre soit toujours présent et que le prix ait toujours une valeur numérique.

Sélectionner une catégorie: Informatique ▼

Sélectionner un livre: Approche fonctionnelle de la programmation ▼

ISBN: 2744070017

ISBN doit être entré sous le format x-xxxx-xxxx-x

Titre: *

Prix: numeric

Auteur:

Ajouter

Cas 1 : l'administrateur peut entrer un nouveau livre dont la catégorie est celle choisie. Cette figure montre que l'utilisateur a entré une valeur pour ISBN qui ne respecte pas le format demandé. Alors un message d'erreur est affiché. Ainsi, * signifie l'absence d'une valeur pour le titre et numeric signifie que la valeur entrée pour le prix n'est pas numérique.

Sélectionner une catégorie: Informatique ▼

Sélectionner un livre: LES RESEAUX ▼

ISBN: 2-7440-7001-7

Titre: LES RESEAUX

Prix: 30

Auteur: Andrew Tanenbaum

Modifier

Supprimer

Cas 2 : l'administrateur peut modifier ou bien supprimer le livre choisi.

Exercice 3 : Page User.aspx

- Cette page est destinée aux utilisateurs de type '*normal*'. Elle leur permet de visualiser tous les livres achetés (le lien voir permet de naviguer vers la page **Historique.aspx**) ou d'acheter des nouveaux livres.

• Dans le second cas, l'utilisateur peut choisir une catégorie parmi la liste. Cette action permet d'afficher dans le premier tableau tous les livres appartenant à la catégorie choisie. Un deuxième tableau contient tous les livres achetés par l'utilisateur **le même jour** où celui-ci consulte le site.

- La dernière colonne du premier tableau contient pour chaque livre un lien vers la **même page (User.aspx)** qui permet à un utilisateur d'acheter un livre donné. Si l'utilisateur clique sur le lien, le livre concerné est ajouté à la base de données (**table LivresAchetes**). Ainsi, ce livre est transporté du premier tableau vers le deuxième tableau.

Liste des livres achetés: [Voir](#)

Sélectionner une catégorie:

ISBN	Titre	Prix	Auteur	
1-4493-1436-8	MongoDB and PHP	15	Steve Francia	Acheter
1-6172-9049-1	Html5 in Action	29	Rob Crowther	Acheter
0-3213-5668-3	Effective Java	34	Joshua Bloch	Acheter
1-4493-1144-1	Programming Interactivity	37	Joshua Noble	Acheter

ISBN	Titre	Categorie	Prix	Auteur
2-7440-7001-7	LES RESEAUX	2	30	Andrew Tanenbaum
0-3216-2321-5	The C++ Standard Library: A Tutorial and Reference	2	65	Nicolai M. Josuttis

Exercice 4 : Historique.aspx

- Cette page affiche tous les livres achetés par un utilisateur donné dans laquelle les informations d'un seul livre sont affichées à la fois. La navigation entre les livres se fait grâce aux deux liens (<< et >>).

