Effectuer des validations simples

par Traduction par Nico-pyright(c) (Page d'accueil)

Date de publication : mai 2009

Dernière mise à jour :

Apprenez comment effectuer la validation dans une application ASP.NET MVC. Dans ce tutoriel, Stephen Walther vous présente le modèle d'état et les méthodes Helper HTML de validation.

Traduction....3
Effectuer des validations simples....3
Comprendre le modèle d'état....3
Utilisation des méthodes Helpers de validation....4
Validation avant Binding et validation après Binding....8
Conclusion....10

Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

Traduction

Cet article est la traduction la plus fidèle possible de l'article original : **Performing Simple Validation**

Effectuer des validations simples

Le but de ce tutoriel est d'expliquer comment vous pouvez effectuer des validations au sein d'une application ASP.NET MVC. Par exemple, vous apprendrez comment empecher une personne de soumettre un formulaire lorsque celui-ci ne possède pas de valeur pour un champ obligatoire. Vous apprendrez à utiliser le modèle d'État et les méthodes Helper HTML de validation.

Comprendre le modèle d'état

Vous utilisez le modèle d'état - ou, plus précisément, le dictionnaire du modèle d'état- pour représenter les erreurs de validation. Par exemple, l'action "créer" *(Create())* du listing 1 valide les propriétés d'une classe produit (*Product*) avant d'ajouter le produit à une base de données.

Je ne recommande pas que vous ajoutiez votre validation ou la logique de votre base de données à un contrôleur. Un contrôleur ne devrait contenir que la logique liée au contrôle de flux de l'application . Nous prenons un raccourci pour garder les choses simples.

Listing 1 - Controllers\ProductController.cs

```
//
// POST: /Product/Create

[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Create([Bind(Exclude="Id")] Product productToCreate)
{
    // Logique de validation
    if (productToCreate.Name.Trim().Length == 0)
        ModelState.AddModelError("Name", "Name is required.");
    if (productToCreate.Description.Trim().Length == 0)
        ModelState.AddModelError("Description", "Description is required.");
    if (productToCreate.UnitsInStock < 0)
        ModelState.AddModelError("UnitsInStock", "Units in stock cannot be less than zero.");
    if (!ModelState.IsValid)
        return View();

    // Logique de la base de données
    try
    {
        _entities.AddToProductSet(productToCreate);
        _entities.SaveChanges();
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

Dans le Listing 1, le nom, la description et le stock unitaire de la classe de Product sont validées. Si l'une de ces propriétés échoue au test de validation, un message d'erreur est ajouté au dictionnaire du modèle d'état (représenté par la propriété **ModelState** de la classe de contrôleur).
S'il ya des erreurs dans le modèle d'état, alors la propriété **ModelState.IsValid** retourne faux (*false*). Dans ce cas, le formulaire HTML permettant la création d'un nouveau produit est réaffichée. Sinon, si il n'y a pas d'erreurs de validation, le nouveau produit est ajouté à la base de données.

Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

Utilisation des méthodes Helpers de validation

Le framework ASP.NET MVC comprend deux méthodes Helper de validation : la méthode Html.ValidationMessage() et la méthode Html.ValidationSummary(). Vous pouvez utiliser ces deux méthodes dans une vue pour afficher les messages d'erreur de validation.
Les méthodes Html.ValidationMessage() et Html.ValidationSummary() sont utilisées dans les vues de création (Create()) et de modification (Edit()) qui sont générées automatiquement par le framework ASP.NET MVC.
Suivez ces étapes pour générer la vue Create :

- 1. Cliquez-droit sur l'action Create() dans le contrôleur Product et sélectionnez l'option de menu **Add View** (Ajouter une vue) (voir Figure 1).
- 2. Dans la boîte de dialogue Add View, cochez la case **Create a strongly-typed view** (Créer une vue fortement typé) (voir Figure 2).
- 3. Dans la liste déroulante **View data class**, sélectionnez la classe Product.
- 4. Dans la liste déroulante **View Content**, sélectionnez Create.
- 5. Cliquez sur le bouton Ajouter.

Assurez-vous d'avoir compilé votre application avant d'ajouter une vue. Sinon, la liste des classes ne figurera pas dans la liste déroulante View data class.

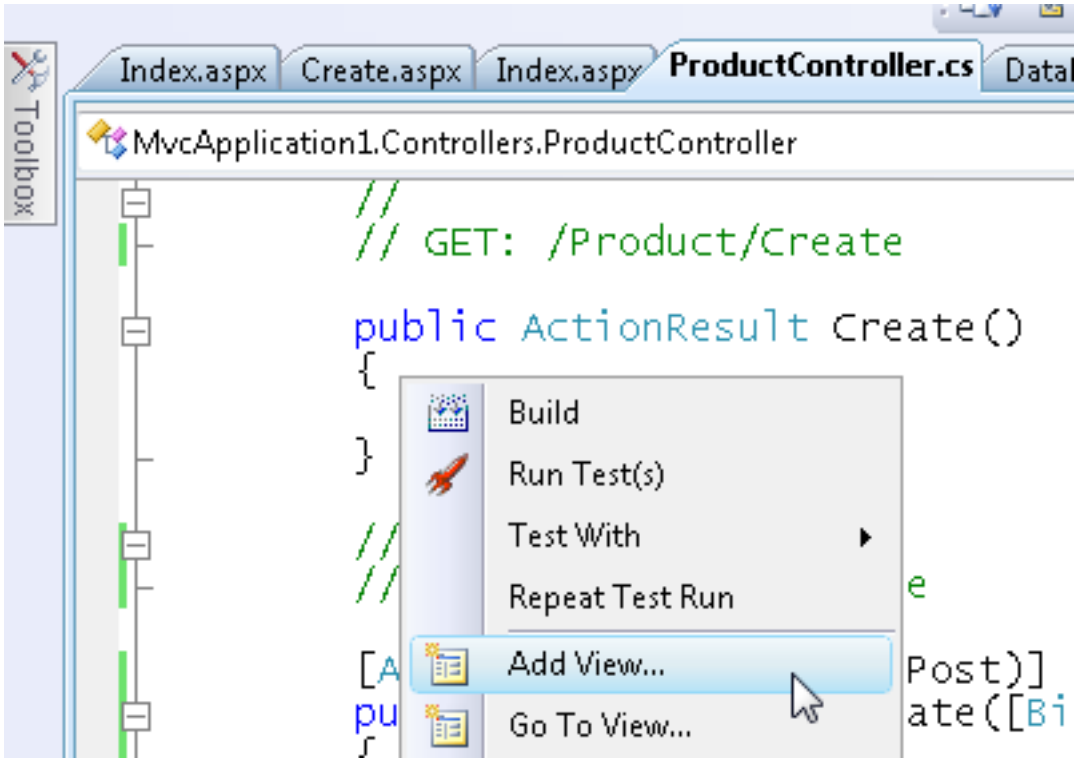

Figure 01 : Ajouter une vue

Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

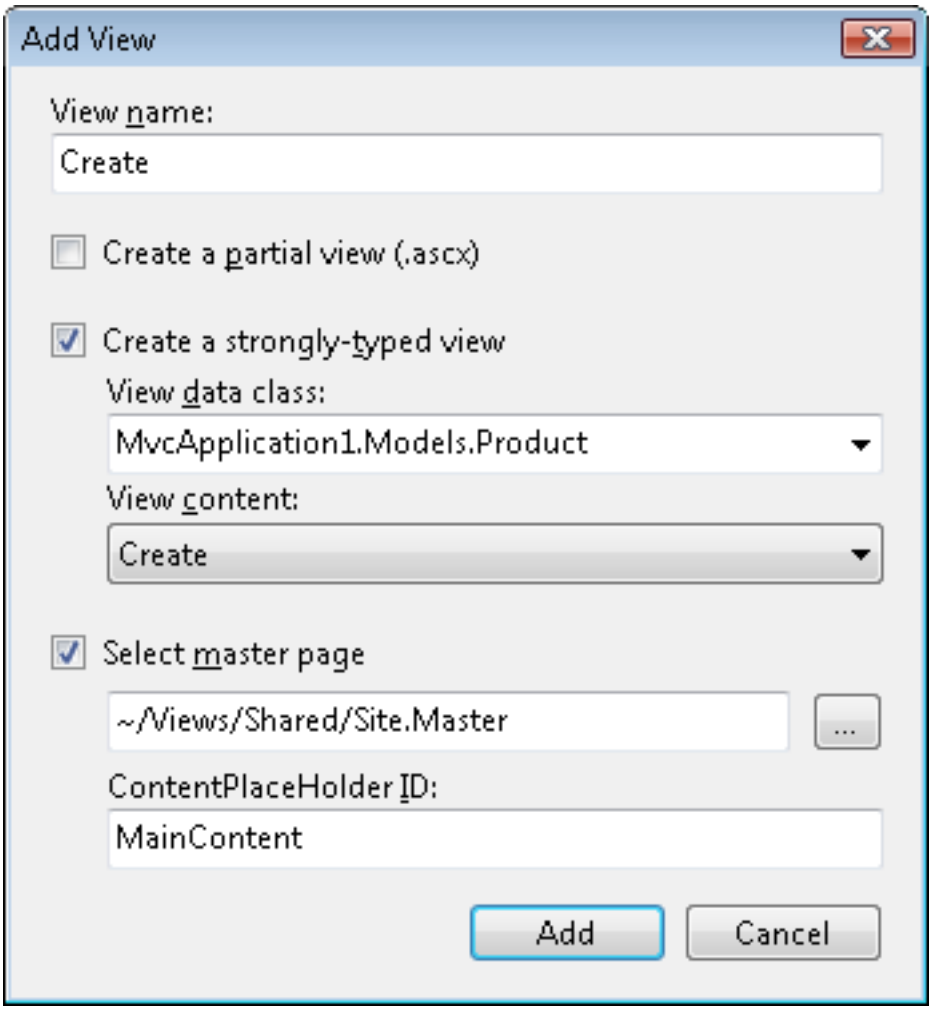

Figure 02 : Créer une vue fortement typée

Après avoir terminé ces étapes, vous obtenez la vue Create : Listing 2

Listing 2 : Views\Product\Create.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
    Inherits="System.Web.Mvc.ViewPage<MvcApplication1.Models.Product>" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
 <title>Create</title>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">

    <h2>Create</h2>

    <%= Html.ValidationSummary() %>

    <% using (Html.BeginForm()) {%>

        <fieldset>
            <legend>Fields</legend>
            <p>
                <label for="Name">Name:</label>
                <%= Html.TextBox("Name") %>
                <%= Html.ValidationMessage("Name", "*") %>
            </p>
            <p>
                <label for="Description">Description:</label>
                <%= Html.TextBox("Description") %>
                <%= Html.ValidationMessage("Description", "*") %>
```


Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

Listing 2 : Views\Product\Create.aspx

```
            </p>
            <p>
                <label for="Price">Price:</label>
                <%= Html.TextBox("Price") %>
                <%= Html.ValidationMessage("Price", "*") %>
            </p>
            <p>
                <label for="UnitsInStock">UnitsInStock:</label>
                <%= Html.TextBox("UnitsInStock") %>
                <%= Html.ValidationMessage("UnitsInStock", "*") %>
            </p>
            <p>
                <input type="submit" value="Create" />
            </p>
        </fieldset>

    <% } %>

    <div>
        <%=Html.ActionLink("Back to List", "Index") %>
    </div>

</asp:Content>
```

Dans le Listing 2, la méthode Html.ValidationSummary() est appelée immédiatement au-dessus du formulaire HTML. Cette méthode est utilisé pour afficher une liste des messages d'erreur de validation. La méthode Html.ValidationSummary() affiche les erreurs dans une liste à puces. La méthode Html.ValidationMessage() est appelée à côté de chacun des champs de formulaire HTML. Cette méthode est utilisée pour afficher un message d'erreur juste à côté d'un champ de formulaire. Dans le cas du listing 2, la méthode Html.ValidationMessage() affiche un astérisque quand il ya une erreur.
On peut voir sur la page de la figure 3 comment sont rendus les messages d'erreur par les méthodes de validation lorsque le formulaire est soumis avec des champs manquant et des valeurs incorrectes.

Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

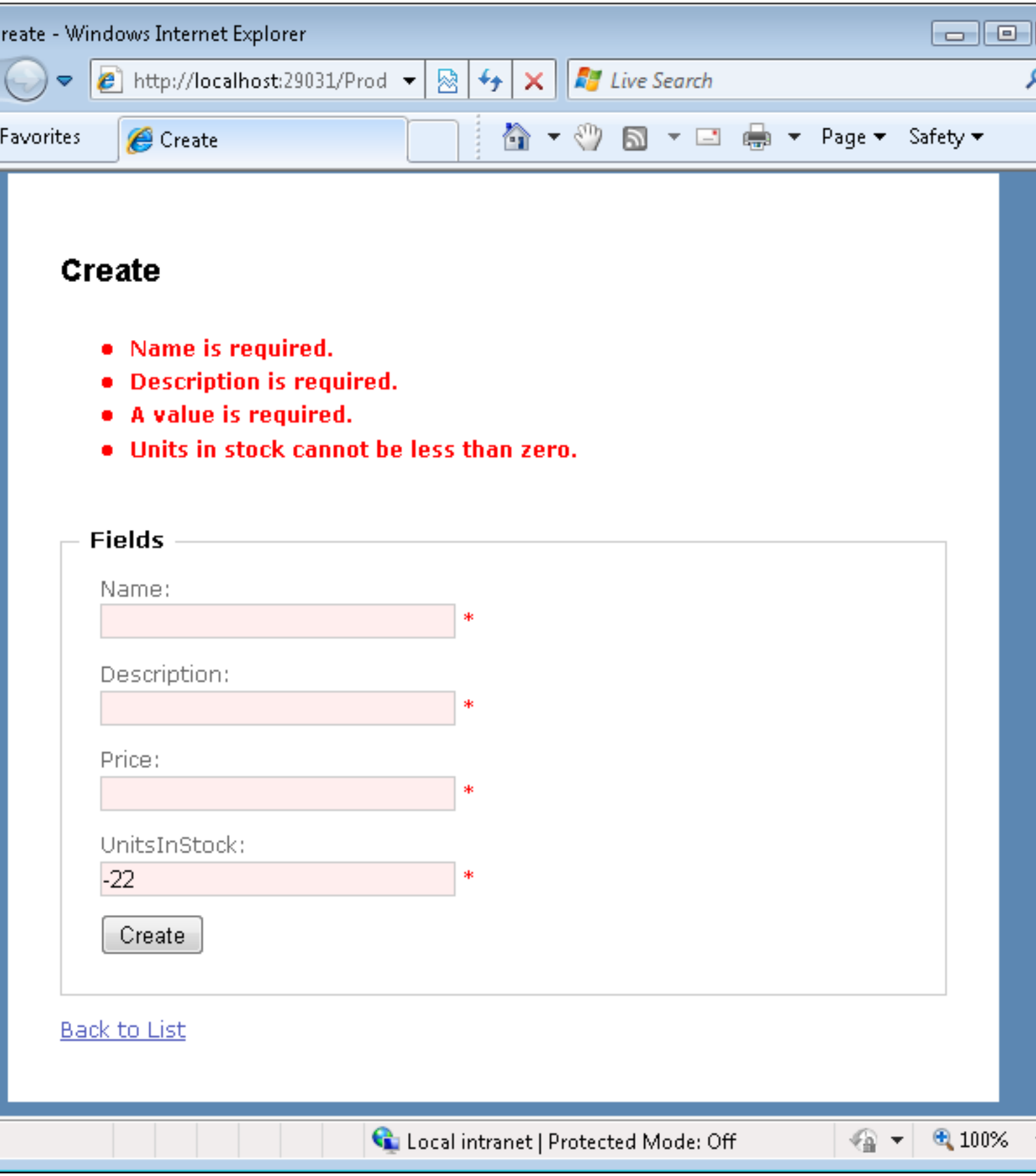

Figure 03 : la vue Create soumis avec des erreurs

Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.
http://dotnet.developpez.com/mvc/mvc-performing-validation/

Notez que l'apparence des champs de saisie de HTML est également modifié quand il y a une erreur de validation. La méthode Helper Html.TextBox() génère un attribut *class="input-validation-error"* quand il y a une erreur de validation associée à la propriété rendu par la méthode Html.TextBox().
Il existe trois classes de feuilles de style en cascade utilisée pour contrôler l'apparence des erreurs de validation:

- input-validation-error appliquée à la balise <input> générée par la méthode Html.TextBox().
- field-validation-error appliquée à la balise générée par la méthode Html.ValidationMessage().
- validation-summary-errors appliquée à la balise <ul> générée par la méthode Html.ValidationSumamry()

Vous pouvez modifier le CSS de ces classes, et donc modifier l'apparence des erreurs de validation, en modifiant le fichier Site.css situé dans le dossier *Content*.

La classe HtmlHelper comprend des propriétés statiques en lecture seule pour récupérer les noms des classes CSS de validation liés. Ces propriétés statiques sont nommées ValidationInputCssClassName, ValidationFieldCssClassName et ValidationSummaryCssClassName.

Validation avant Binding et validation après Binding

Si vous soumettez le formulaire HTML permettant la création d'un produit, et que vous entrez une valeur non valide pour le champ Price et aucune valeur pour le champ UnitsInStock, alors vous aurez les messages de validation que l'on peut voir sur la figure 4. D'où viennent ces messages d'erreur de validation ?

Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

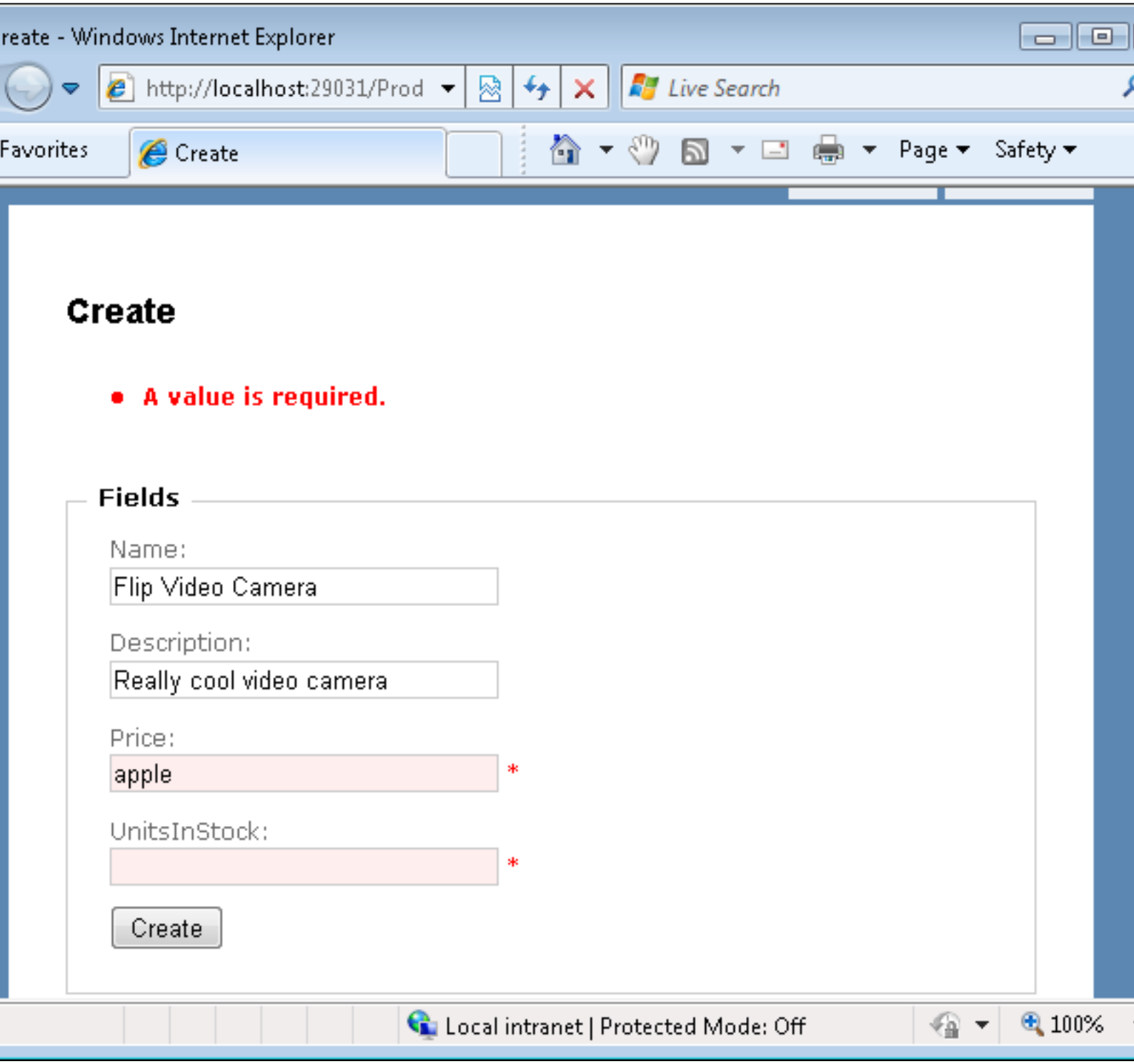

Figure 04 : Erreur de validation avant Binding

Il ya en fait deux types de messages d'erreur de validation - ceux générés avant que les champs de formulaire HTML soient liés à une classe et ceux générés après que les champs du formulaire soient liés à la classe. En d'autres termes, il y a des erreurs de validation avant binding (PreBinding) et des erreurs de validation après binding (PostBinding). L'action Create() exposée par le contrôleur Product dans le listing 1 accepte une instance de la classe Product. La signature de la méthode Create ressemble à ceci:

Listing 1 :

```
public ActionResult Create([Bind(Exclude="Id")] Product productToCreate)
```


Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

Les valeurs des champs de formulaire HTML sont liées à l'objet productToCreate par ce qu'on appelle un modèle de liaison (*model binder*). Le modèle de liaison par défaut ajoute un message d'erreur au modèle d'état automatiquement quand il ne peut pas lier un champ de formulaire à une propriété de formulaire.
Le modèle de liaison par défaut ne peut pas lier la chaine "pomme" à la propriété Price de la classe Product. Vous ne pouvez pas affecter une chaîne à une propriété décimale. Par conséquent, le modèle de liaison ajoute une erreur au modèle d'état.
Le modèle de liaison par défaut ne peut pas non plus attribuer une valeur nulle à une propriété qui ne l'accepte pas. En particulier, le modèle de liaison ne peut pas attribuer une valeur nulle à la propriété UnitsInStock. Une fois de plus, le modèle de liaison abandonne sa tentative de liaison et ajoute un message d'erreur au modèle d'État.
Si vous souhaitez personnaliser l'apparence de ces messages d'erreur avant validation, vous devez créer des chaînes de ressources pour ces messages.

Conclusion

Le but de ce tutoriel est de décrire la mécanique de base de la validation du framework ASP.NET MVC. Vous avez appris à utiliser le modèle d'Etat et les méthodes Helper de validation HTML. Nous avons également discuté de la distinction entre validation avant binding validation et après binding. Dans d'autres tutoriels, nous discuterons de diverses stratégies pour déplacer votre code de validation de vos contrôleurs vers vos classes de modèle.

Copyright 2009 - Equipe dotnet. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.