

La validation sous Flex

par Olivier Bugalotto ([Mes articles](#))

Date de publication : 20/08/2007

Ce tutoriel vous expliquera comment mettre en place un processus de validation dans vos formulaires.

- I - Introduction
- II - Les validateurs Flex
- III - Ajout des validateurs
- IV - Déclenchement de la validation
- V - Utilisation de la classe Validator
- VI - Nettoyage des messages d'erreurs
- VII - Les évènements VALID et INVALID
- VIII - Personnaliser les messages d'erreur
- IX - Création d'un composant de validation
- X - Conclusion
- XI - Les sources

I - Introduction

Le développement de formulaire de saisie demande aussi la mise en place d'une logique de validation des données entrées par l'utilisateur pour garantir leur sécurité. Cette logique peut parfois être longue et fastidieuse.

Voici un exemple de validation :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
width="380" height="250" verticalAlign="middle">

<mx:Script>
<![CDATA[
import mx.controls.Alert;
private function onConnection():void {
var email:String = txtEmail.text;
var pass:String = txtPass.text;

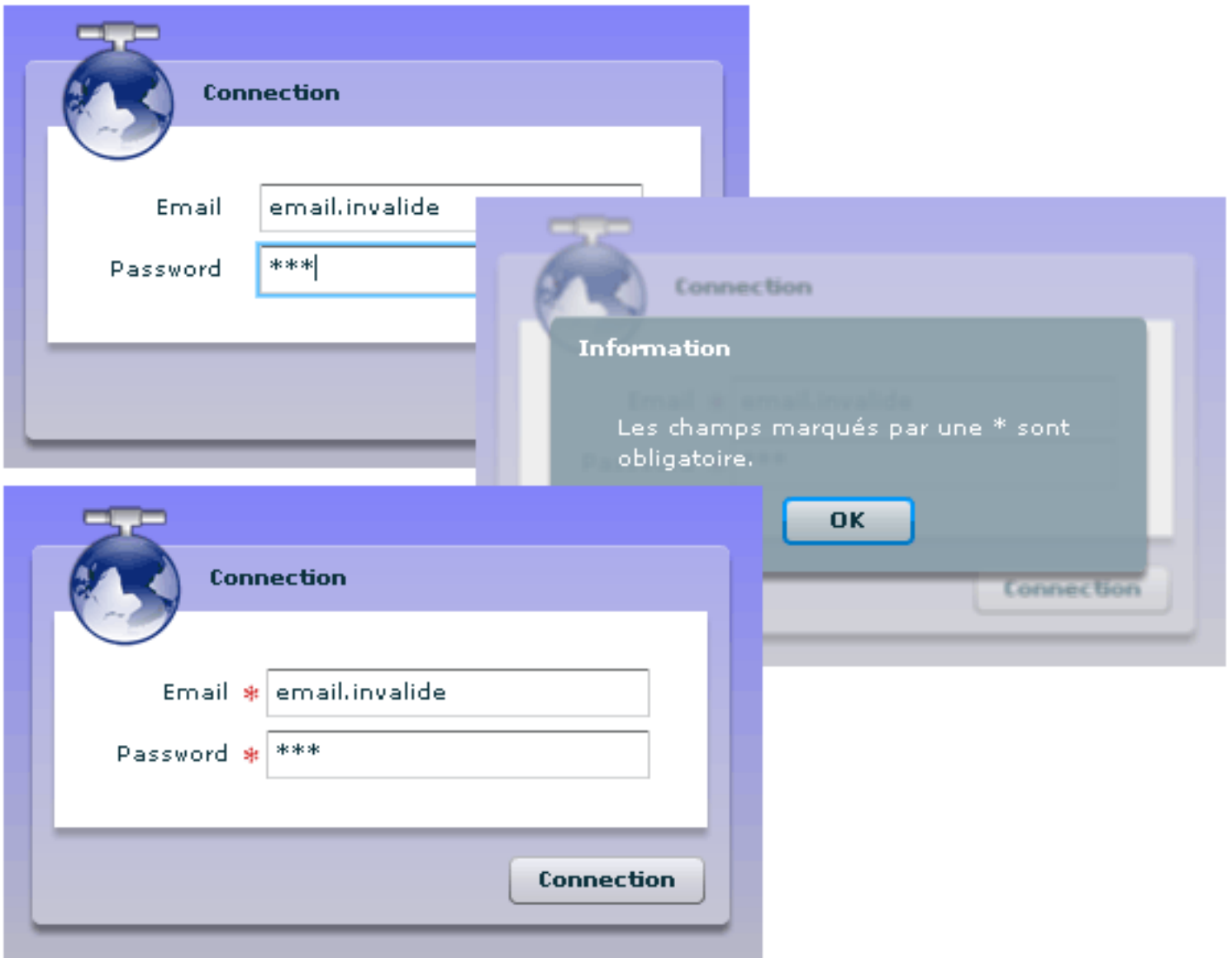
var isEmail:Boolean = validateEmail(email);
var isEmpty:Boolean = pass.length != 0;

if(isEmail && isEmpty) {
frmItemEmail.required = frmItemPass.required = false;
Alert.show("Validation reussie!", "Information");
} else {
frmItemEmail.required = frmItemPass.required = true;
Alert.show("Les champs marques par une * sont obligatoire.",
"Information");
}
}

private function validateEmail(email:String):Boolean {
var pattern:RegExp = /(\\w|[_\\.\\-])+@((\\w|-)+\\.)+\\w{2,4}+\\/;
var result:Object = pattern.exec(email);
if(!result)
return false;
return true;
}
]]>
</mx:Script>

<mx:Panel width="316" height="172" layout="absolute" title="Connection">
<mx:Form right="10" left="10" top="10" bottom="10">
<mx:FormItem id="frmItemEmail" label="Email" width="100%">
<mx:TextInput id="txtEmail" width="100%"/>
</mx:FormItem>
<mx:FormItem id="frmItemPass" label="Password" width="100%">
<mx:TextInput id="txtPass" width="100%" displayAsPassword="true"/>
</mx:FormItem>
</mx:Form>
<mx:ControlBar height="42" y="128" horizontalAlign="right">
<mx:Button label="Connection" click="onConnection()"/>
</mx:ControlBar>
</mx:Panel>

</mx:Application>
```



Comme vous pouvez le remarquer la logique de validation est tout de même importante pour valider deux champs de saisies. Imaginez la logique pour des formulaires plus importants comme l'inscription d'un utilisateur.

II - Les validateurs Flex

Le framework de Flex 2 propose un ensemble de composants non visuels nous permettant la validation des données. Ces composants sont appelés des « validateurs »; il en existe plusieurs actuellement :

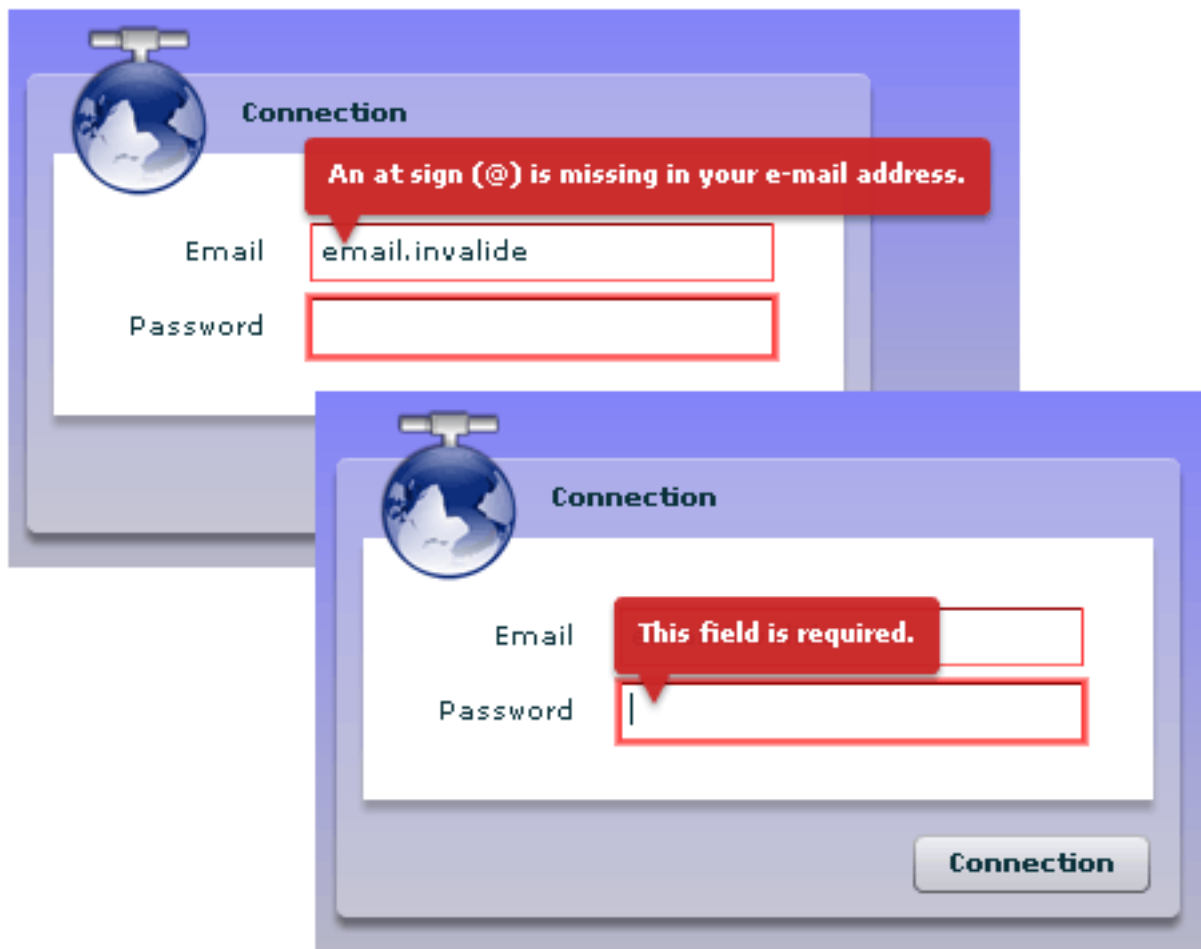
Composants de validation	Description
StringValidator	Composant de validation pour les chaînes de caractères
NumberValidator	Pour les nombres entiers et à virgules
CurrencyValidator	Pour valider les valeurs monétaires
DateValidator	Pour valider les dates
EmailValidator	Pour valider une adresse électronique
ZipCodeValidator	Pour valider un code postal
PhoneNumberValidator	Pour les numéros de téléphone
CreditCardValidator	Pour les numéros de carte de crédit
SocialSecurityValidator	Pour les numéros de sécurité sociale
RegExpValidator	Pour les numéros de sécurité sociale

III - Ajout des validateurs

Voici donc notre exemple précédent avec l'emploi des validateurs :

```
<mx:EmailValidator id="validEmail" source="{txtEmail}" property="text" />
<mx:StringValidator id="validPass" source="{txtPass}" property="text" />

<mx:Panel width="316" height="172" layout="absolute" title="Connection">
<mx:Form right="10" left="10" top="10" bottom="10">
  <mx:FormItem label="Email" width="100%">
    <mx:TextInput id="txtEmail" width="100%" />
  </mx:FormItem>
  <mx:FormItem label="Password" width="100%">
    <mx:TextInput id="txtPass" width="100%" displayAsPassword="true" />
  </mx:FormItem>
</mx:Form>
<mx:ControlBar height="42" y="128" horizontalAlign="right">
  <mx:Button id="btnConnect" label="Connection" click="onConnection()" />
</mx:ControlBar>
</mx:Panel>
```



Le gros problème avec cette exemple c'est qu'il n'empêche pas le « click » sur le bouton « connection ».

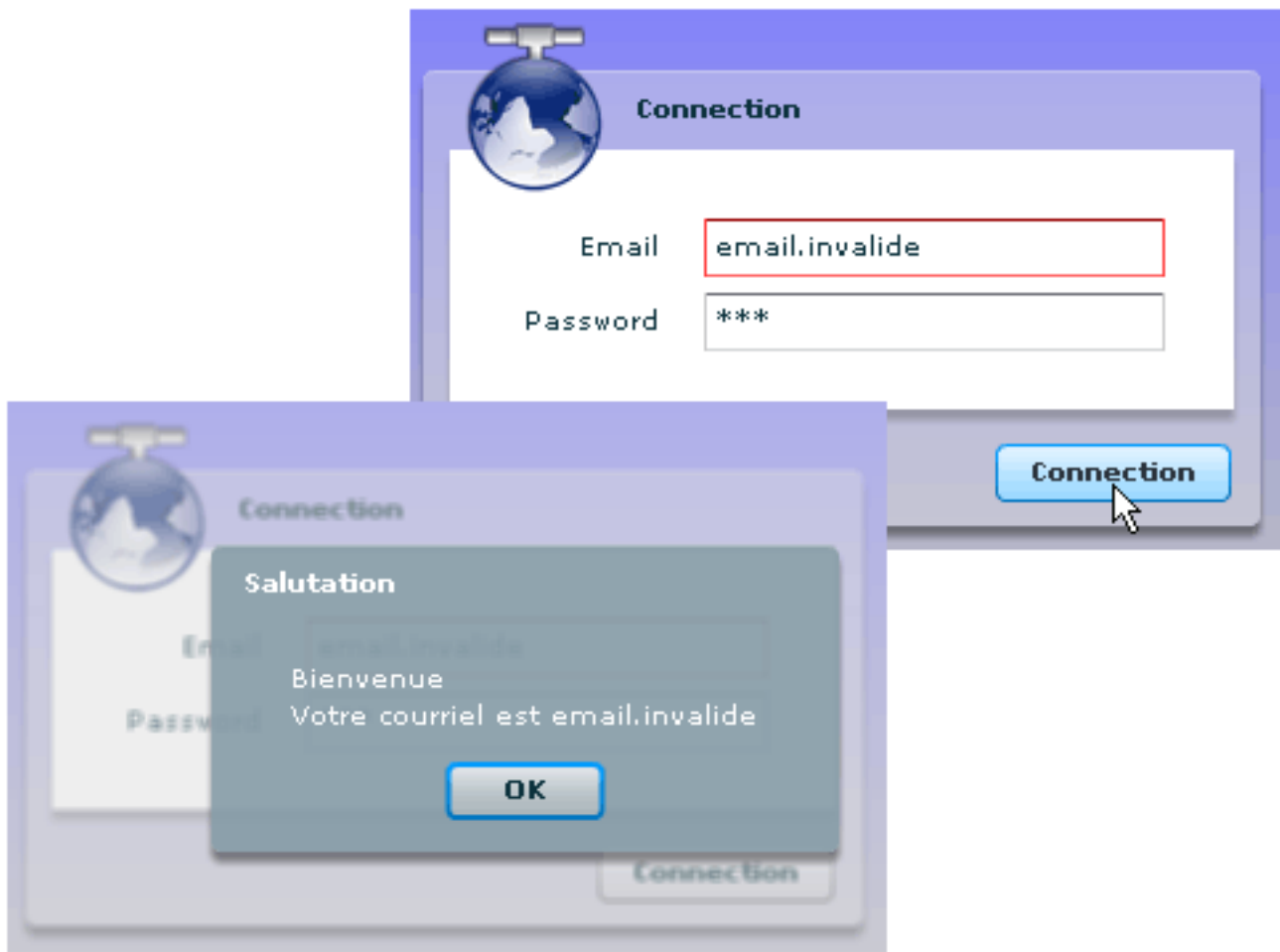
IV - Déclenchement de la validation

Les validateurs déclenchent la validation à la perte du focus du composant à valider.

i Ce n'est pas réellement sur l'évènement `focusOut` mais sur `valueCommit` que se produit la validation.

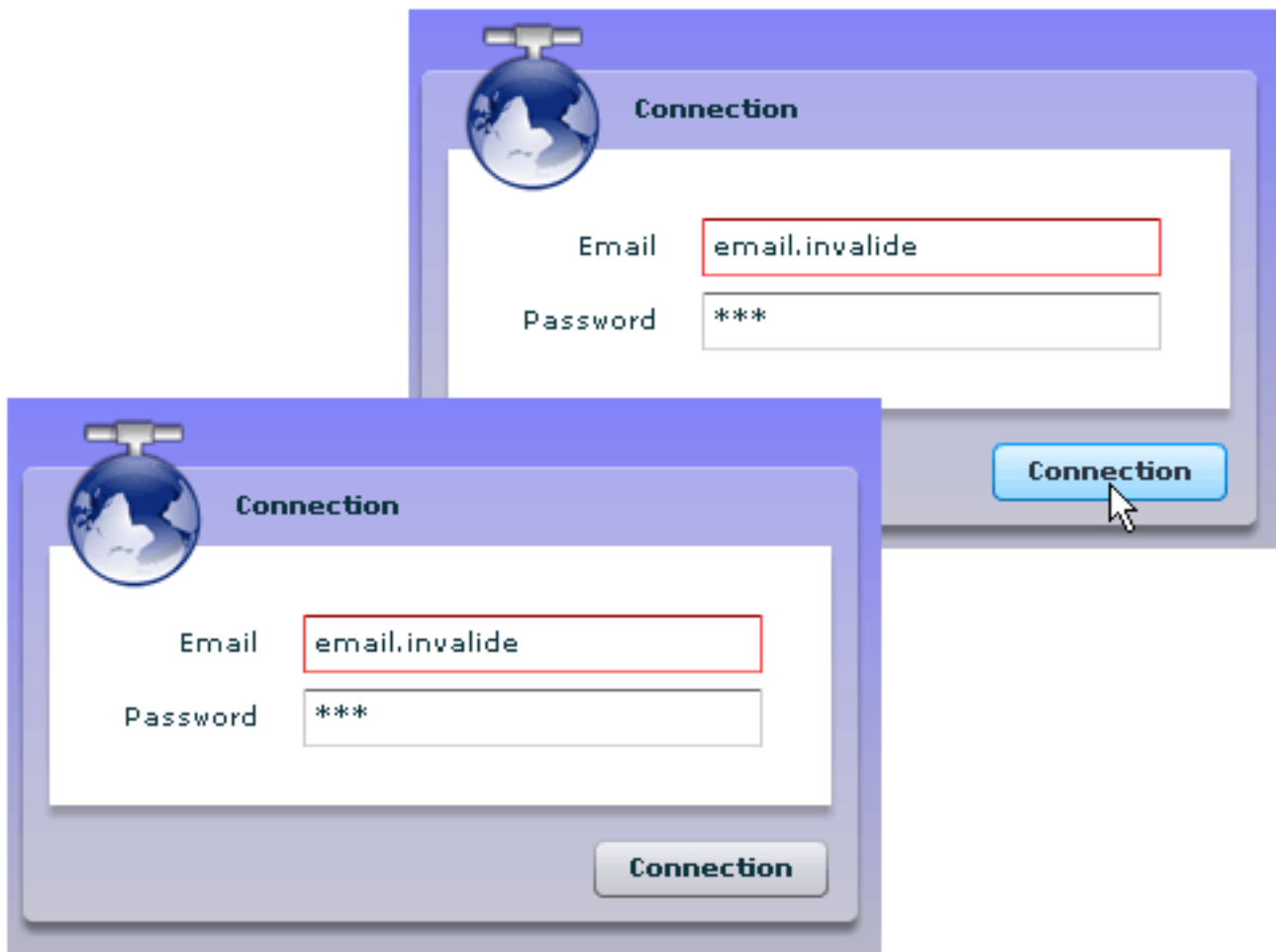
Il nous faut modifier ce comportement et déclencher la validation sur l'action du « click » du bouton. Il suffit pour cela de modifier la propriété `triggerEvent` mais aussi la propriété `trigger` du validateur :

```
<mx:EmailValidator id="validEmail" source="{txtEmail}" property="text"
  trigger="{btnConnect}" triggerEvent="click" />
<mx:StringValidator id="validPass" source="{txtPass}" property="text"
  trigger="{btnConnect}" triggerEvent="click" />
```



Cela ne résout pas le problème, nous avons bien la validation sur le « click » du bouton mais cela n'empêche pas l'affichage de la fenêtre d'alerte. Pour résoudre ce problème, il suffit d'exécuter les méthodes `validate()` des validateurs qui vont nous retourner un événement que nous testerons :

```
private function onConnection():void {  
    var email:String = txtEmail.text;  
    var pass:String = txtPass.text;  
  
    var reValidEmailEvent:ValidationResultEvent = validEmail.validate();  
    var reValidPassEvent:ValidationResultEvent = validPass.validate();  
  
    // Nous testons sur l'evenement retourne par la methode validate()  
    if(reValidEmailEvent.type == ValidationResultEvent.VALID  
        && reValidPassEvent.type == ValidationResultEvent.VALID) {  
        // La methode StringUtil.substitute permet d'utiliser des indices  
        // pour les substituer par les parametres :  
        // ici {0} sera substitue par la valeur de la variable "email"  
        var msg:String = StringUtil.substitute("Bienvenue\nVotre courriel est {0}",email);  
        Alert.show(msg,"Salutation");  
    }  
}
```



V - Utilisation de la classe Validator

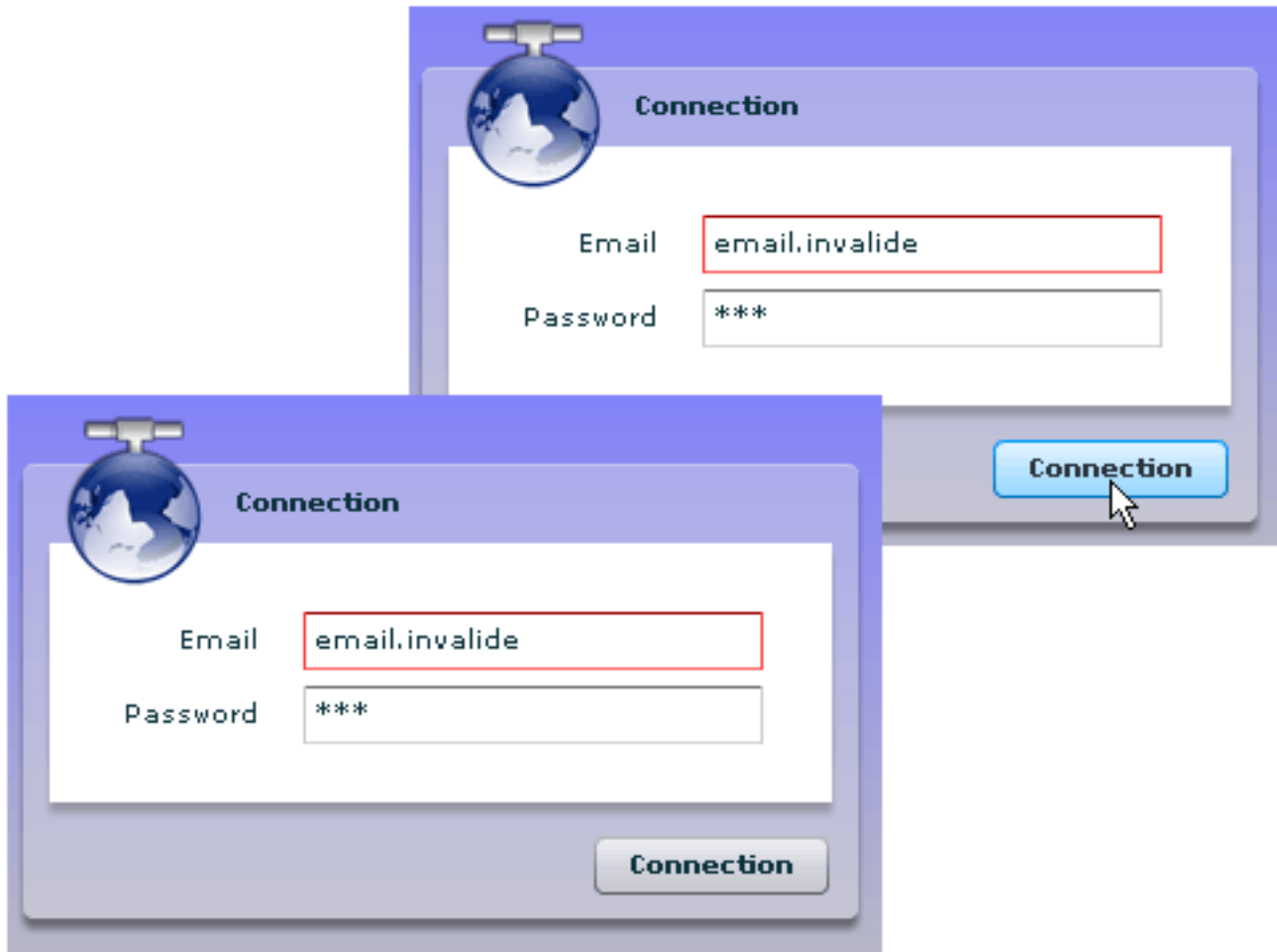
En y regardant de plus près, la logique pour valider les champs reste encore lourde dans le cas d'un long formulaire. Il faudrait pouvoir valider le tout en une seule fois et bien cela est possible avec la classe *Validator* ; la classe *Validator* est la classe de base pour les validateurs prédéfinis.

Cette classe propose la méthode statique et très pratique *validateAll()* qui valide un tableau de validateurs et renvoie un tableau vide si la validation a réussi :

```
private function onConnection():void {
    var email:String = txtEmail.text;
    var pass:String = txtPass.text;

    var validators:Array = Validator.validateAll([validEmail,validPass]);

    // Si le tableau contient au moins un element
    // cela indique une erreur
    if(!validators.length) {
        // La methode StringUtil.substitute permet d'utiliser des indices
        // pour les substituer par les parametres :
        // ici {0} sera substitué par la valeur de la variable "email"
        var msg:String = StringUtil.substitute("Bienvenue\nVotre courriel est {0}",email);
        Alert.show(msg, "Salutation");
    }
}
```

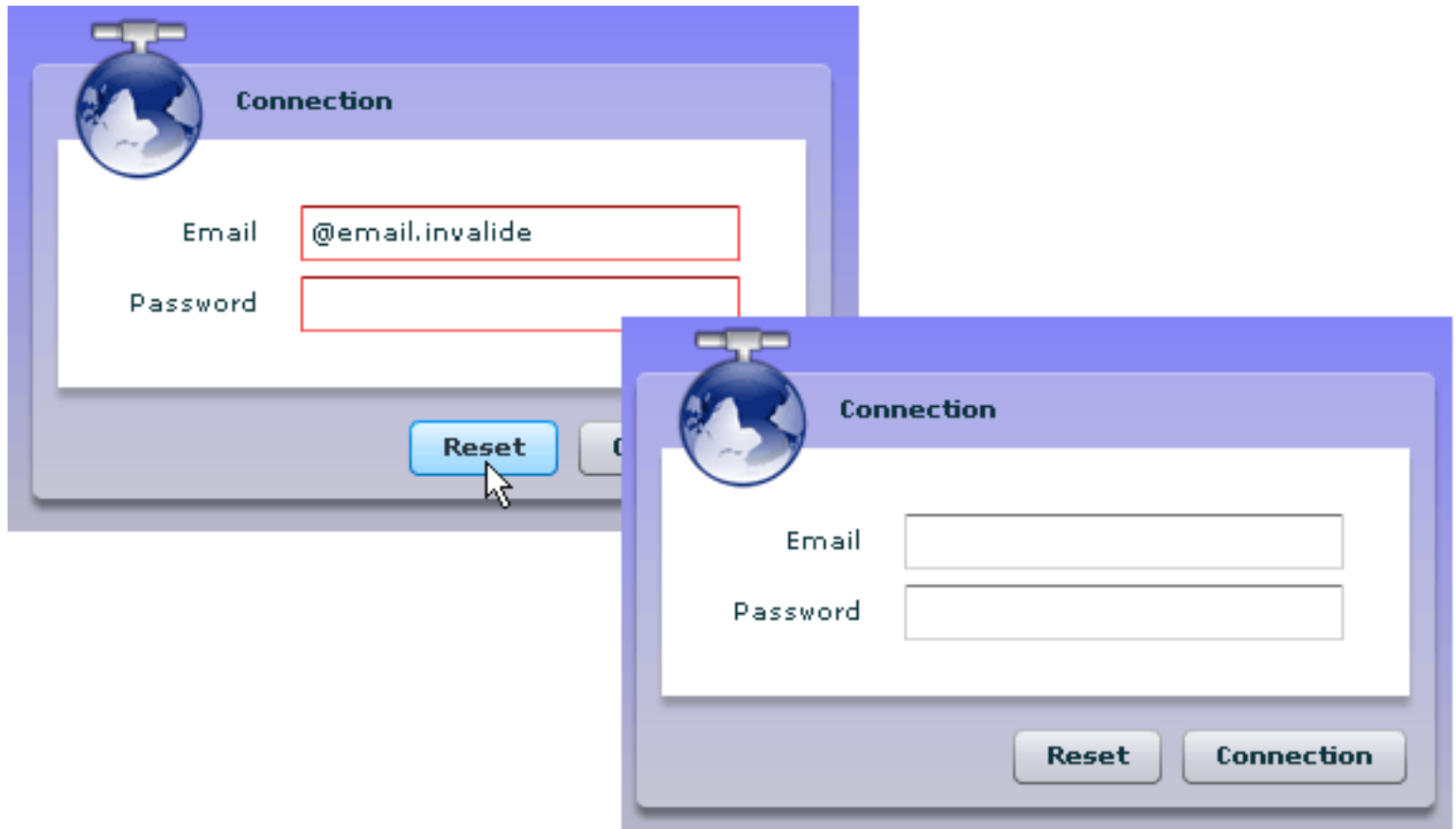


Même résultat, mais code plus pratique

Dans le cas où le tableau contient au moins un élément, ce dernier est un élément de type *ValidationResultEvent*.

VI - Nettoyage des messages d'erreurs

Nous pouvons aussi nettoyer les messages d'erreurs en modifiant la propriété `errorString` des composants à valider :



Nettoyage des messages d'erreurs

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
verticalAlign="middle" width="450" height="250" horizontalAlign="left"
backgroundGradientColors="[#8080ff, #c0c0c0]">

<mx:Script>
<![CDATA[
import mx.validators.Validator;
import mx.utils.StringUtil;
import mx.controls.Alert;
import mx.events.ValidationResultEvent;

private function onConnection():void {
var email:String = txtEmail.text;
var pass:String = txtPass.text;

var validators:Array = Validator.validateAll([validEmail,validPass]);

if(!validators.length) {
var msg:String = StringUtil.substitute(
"Bienvenue\nVotre courriel est {0}",email);
Alert.show(msg,"Salutation");
}
}
]]>
</mx:Script>
</mx:Application>
```

```
}

private function onResetForm():void {
    // Suppression du nom d'utilisateur saisi
    txtEmail.text = "";
    // Suppression du message d'erreur
    txtEmail.errorString = "";

    txtPass.text = "";
    txtPass.errorString = "";
}
]]>
</mx:Script>

<mx:EmailValidator id="validEmail" source="{txtEmail}" property="text"
    trigger="{btnConnect}" triggerEvent="click"
    requiredFieldError="Ce champ est obligatoire."
    missingAtSignError="Le caractere @ est manquant dans votre adresse email."
    missingPeriodInDomainError="Le domaine est manquant dans votre adresse email."
    missingUsernameError="Le nom d'utilisateur est manquant dans votre adresse email." />
<mx:StringValidator id="validPass" source="{txtPass}" property="text"
    trigger="{btnConnect}" triggerEvent="click"
    requiredFieldError="Ce champ est obligatoire." />

<mx:Panel width="316" height="172" layout="absolute" title="Connection"
    titleIcon="@Embed('assets/network.png')">
    <mx:Form right="10" left="10" top="10" bottom="10">
        <mx:FormItem label="Email" width="100%">
            <mx:TextInput id="txtEmail" width="100%" />
        </mx:FormItem>
        <mx:FormItem label="Password" width="100%">
            <mx:TextInput id="txtPass" width="100%" displayAsPassword="true" />
        </mx:FormItem>
    </mx:Form>
    <mx:ControlBar height="42" y="128" horizontalAlign="right">
        <mx:Button label="Reset" click="onResetForm()" />
        <mx:Button id="btnConnect" label="Connection" click="onConnection()" />
    </mx:ControlBar>
</mx:Panel>

</mx:Application>
```

VII - Les évènements VALID et INVALID

Nous pouvons gérer la validation en écoutant les évènements *valid* et *invalid* émis par le composant de validation ou par le composant de saisie lui-même.

Dans cet exemple, nous écoutons les deux évènements sur le composant de validation pour activer ou non le bouton d'enregistrement :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
  verticalAlign="middle">

  <mx:Script>
    <![CDATA[
      import mx.controls.Alert;
      import mx.utils.StringUtil;
      private function onValid():void {
        btnSave.enabled = true;
      }

      private function onInvalid():void {
        btnSave.enabled = false;
      }

      private function onSave():void {
        var email:String = txtEmail.text;
        var nom:String = txtNom.text ? txtNom.text : "";
        var prenom:String = txtPrenom.text ? txtPrenom.text : "";

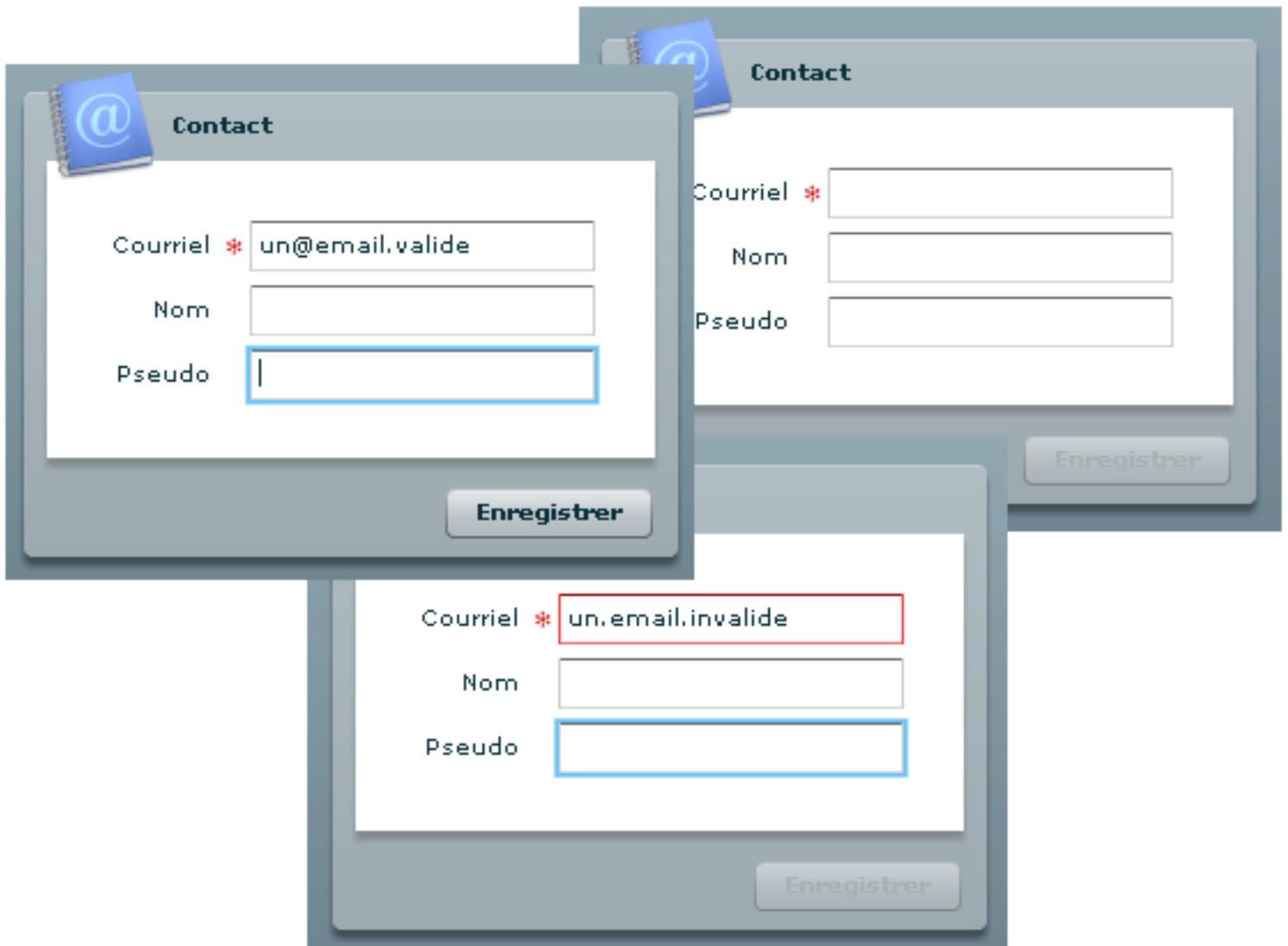
        var msg:String = StringUtil.substitute("Courriel: {0}\nNom: " +
          "{1}\nPrÃ©nom: {2}", email, nom, prenom);

        Alert.show(msg, "Nouveau contact");
      }
    ]]>
  </mx:Script>

  <mx:EmailValidator source="{txtEmail}" property="text"
    valid="onValid()" invalid="onInvalid()" />

  <mx:Panel title="Contact" width="284" height="202" layout="absolute"
    titleIcon="@Embed('assets/addressbook2.png')">
    <mx:Form right="10" left="10" top="10" bottom="10">
      <mx:FormItem label="Courriel" width="100%" required="true">
        <mx:TextInput id="txtEmail" width="100%" />
      </mx:FormItem>
      <mx:FormItem label="Nom" width="100%">
        <mx:TextInput id="txtNom" width="100%" />
      </mx:FormItem>
      <mx:FormItem label="Pseudo" width="100%">
        <mx:TextInput id="txtPrenom" width="100%" />
      </mx:FormItem>
    </mx:Form>
    <mx:ControlBar height="41" y="143" horizontalAlign="right">
      <mx:Button id="btnSave" label="Enregistrer" enabled="false"
        click="onSave()" />
    </mx:ControlBar>
  </mx:Panel>

</mx:Application>
```



Activation du bouton d'enregistrement

VIII - Personnaliser les messages d'erreur

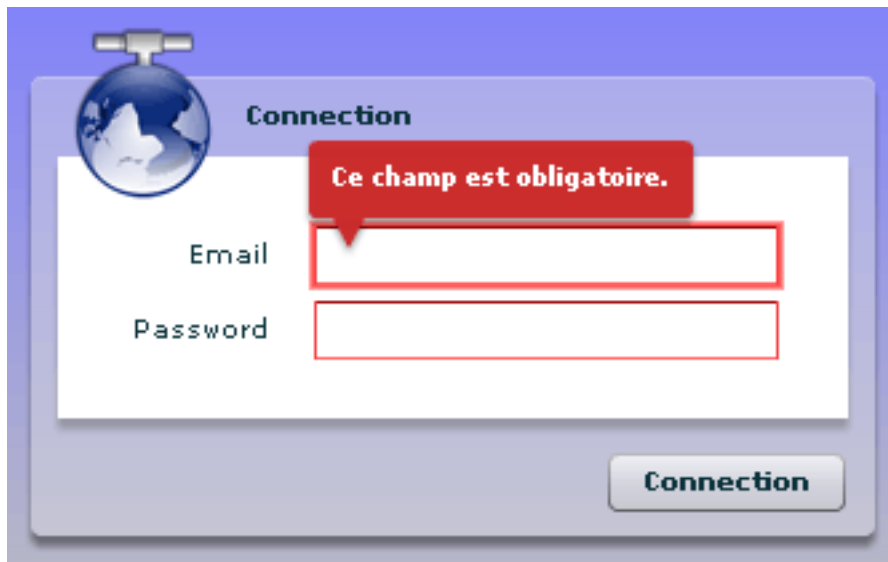
Nous allons voir comment personnaliser les messages d'erreurs qui jusqu'à maintenant été définis par défaut et qui plus est en anglais.

La classe de base *Validator* possède la propriété *requiredFieldError* qui permet de modifier les messages d'erreur.

```

<mx:EmailValidator id="validEmail" source="{txtEmail}" property="text"
  trigger="{btnConnect}" triggerEvent="click"
  requiredFieldError="Ce champ est obligatoire." />
<mx:StringValidator id="validPass" source="{txtPass}" property="text"
  trigger="{btnConnect}" triggerEvent="click"
  requiredFieldError="Ce champ est obligatoire." />

```



Message personnalisé

Mais chaque validateur propose différents messages d'erreur, comme c'est le cas ici avec *EmailValidator* :

```

<mx:EmailValidator id="validEmail" source="{txtEmail}" property="text"
  trigger="{btnConnect}" triggerEvent="click"
  requiredFieldError="Ce champ est obligatoire."
  missingAtSignError="Le caractere @ est manquant dans votre adresse email."
  missingPeriodInDomainError="Le domaine est manquant dans votre adresse email."
  missingUsernameError="Le nom d'utilisateur est manquant dans votre adresse email." />
<mx:StringValidator id="validPass" source="{txtPass}" property="text"
  trigger="{btnConnect}" triggerEvent="click"
  requiredFieldError="Ce champ est obligatoire." />

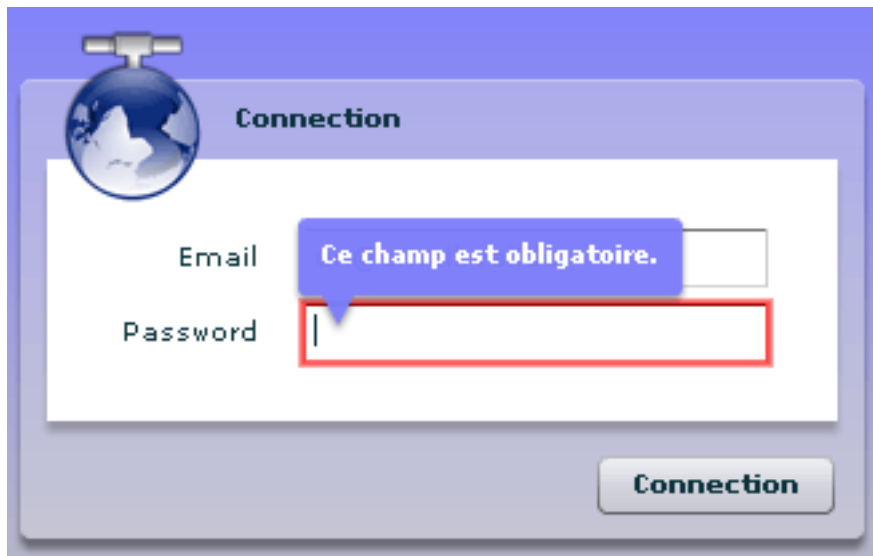
```



Différents messages d'erreurs du validateur EmailValidator

Nous pouvons aussi personnaliser le style de la bulle d'erreur. Pour cela, il suffit de modifier le style `.errorTip` défini par défaut dans la feuille de style `default.css` que vous trouverez dans le répertoire du Flex SDK 2; dans notre exemple, nous allons simplement modifier la couleur de la bulle :

```
<mx:Style>
  .errorTip {
    borderColor: #8080ff;
  }
</mx:Style>
```



The image shows a login form titled "Connection" with a globe icon. It contains two input fields: "Email" and "Password". The "Email" field is empty and has a blue error message bubble pointing to it that says "Ce champ est obligatoire." (This field is mandatory). The "Password" field is also empty and is highlighted with a red border. A "Connection" button is located at the bottom right of the form.

Apparence de la bulle d'erreur personnalisée

IX - Création d'un composant de validation

Pour construire un nouveau composant de validation, il faut créer une nouvelle classe qui hérite de *Validator* et redéfinir la méthode *doValidation()*. C'est cette méthode qui est responsable de la validation.

Nous allons créer un composant de validation qui va comparer deux champs de saisie comme c'est souvent le cas lorsque nous faisons confirmer un mot de passe :

```
package {
import mx.validators.Validator;
import mx.utils.ObjectUtil;
import mx.validators.ValidationResult;
import mx.utils.StringUtil;

public class CompareStringValidator extends Validator {
private var results:Array;

public var valueCompare:Object;

public function CompareStringValidator() {
super();
}

protected override function doValidation(value:Object):Array {
// Initialise le tableau des erreurs
results = [];

results = super.doValidation(value);

// Si le tableau contient au moins un Ã©lÃ©ment
// nous le retournons
if (results.length > 0)
return results;

var strValue:String = String(value);
var compareStrValue:String = String(valueCompare);

// Si ce n'est pas Ã©gale stringCompare nous retourne 0
var result:int = ObjectUtil.stringCompare(strValue,compareStrValue);
if(result) {
var errorMessage:String =
StringUtil.substitute("La valeur {0} n'est pas Ã©gale Ã la " +
"valeur {1}",strValue,compareStrValue);
// Ici, nous construisons un evenement ValidationResult
// pour l'ajouter au tableau
results.push(new ValidationResult(true,null,
"Comparaison invalide",errorMessage));
}

// Nous retournons le tableau
return results;
}
}
```

Et maintenant voici son utilisation dans votre application :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical" xmlns="*">
```

```
<!-- l'attribut (propriete) sourceCompare est la valeur de comparaison -->
<CompareStringValidator source="{txtComparePass}" property="text" sourceCompare="{txtPass.text}"
/>

<mx:Panel width="340" height="207" layout="absolute" title="Inscription">
<mx:Form right="10" left="10" top="10" bottom="10">
  <mx:FormItem label="Email" width="100%">
    <mx:TextInput width="100%" />
  </mx:FormItem>
  <mx:FormItem label="Password" width="100%">
    <mx:TextInput id="txtPass" width="100%" displayAsPassword="true" />
  </mx:FormItem>
  <mx:FormItem label="Confirm password" width="100%">
    <mx:TextInput id="txtComparePass" width="100%" displayAsPassword="true" />
  </mx:FormItem>
</mx:Form>
<mx:ControlBar horizontalAlign="right">
  <mx:Button label="Enregistrer" />
</mx:ControlBar>
</mx:Panel>

</mx:Application>
```

The screenshot shows a registration form titled "Inscription" with a globe icon. It contains three input fields: "Email" (un@email.valide), "Password" (with a red error message: "La valeur pwd2 n'est pas égale à la valeur pwd1."), and "Confirm password" (****). An "Enregistrer" button is located at the bottom right.

Exemple de composant validateur

X - Conclusion

Les composants de validation apportent un certain confort mais nous sentons qu'il y a encore un travail à faire comparé à la validation que nous pouvons trouver dans d'autres langages.

XI - Les sources

Vous trouverez le code source de ce tutorial ici : **Validators.zip (9.72 Mo) (Miroir)**

