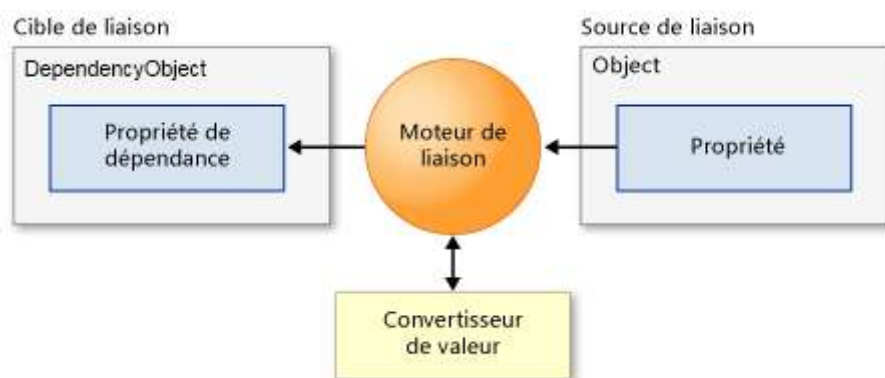


# Feuille de rappel sur les liaisons (bindings) en WPF

Une version mise à jour de ce document est à cette adresse : <http://bindings.wpf-france.fr>



**Rappel sur les liaisons:** Une liaison de données est effectuée entre une cible, une DependencyProperty d'un DependencyObject (par exemple la propriété Text d'un TextBox), et une source de données qui peut être n'importe quel objet CLR. La valeur peut être convertie de la source vers la cible par un convertisseur de valeurs. La liaison est effectuée par le moteur de liaison (binding engine) de WPF.

Liaisons simples	
{Binding}	Liaison sur le DataContext courant
{Binding Nom} ou {Binding Path=Nom}	Liaison sur la propriété « Nom » du DataContext courant
{Binding Nom.Propriete}	Liaison sur la propriété « Propriete » de la propriété « Nom » du DataContext courant
{Binding ElementName=unAutreControle, Path=Propriete }	Liaison sur la propriété « Propriete » de l'élément XAML ayant comme nom « unAutreControle »
{Binding Source=unObjetSource, Path=Propriete }	Liaison sur la propriété « Propriete » de l'objet source « unObjetSource »

**Rappel sur le DataContext :** si la propriété DataContext de l'élément cible est fixée alors on utilise celle-ci. Dans le cas contraire, on utilise le DataContext du premier élément parent dont cette propriété est fixée. Dans le cas où les MarkupExtensions « Source », « ElementName » ou « RelativeSource » ne sont pas utilisées dans les liaisons alors celles-ci s'effectuent sur le DataContext de l'élément cible.

Liaisons relatives (RelativeSource : MSDN)	
{Binding RelativeSource={RelativeSource=Self} }	Liaison sur l'élément cible : l'élément sur lequel on effectue le Binding
{Binding RelativeSource={RelativeSource=Self}, Path=Propriete }	Liaison sur la propriété « Propriete » de l'élément cible

{Binding RelativeSource={RelativeSource=FindAncestor}, AncestorType={x:Type TypeRecherche}, Path=Propriete }	Liaison sur la propriété « Propriete » du premier ancêtre de type « TypeRecherche » de l'élément cible
{Binding RelativeSource={RelativeSource=FindAncestor}, AncestorType={x:Type TypeRecherche}, AncestorLevel=2 Path=Propriete }	Liaison sur la propriété « Propriete » du second ancêtre de type « TypeRecherche » de l'élément cible
{Binding RelativeSource={RelativeSource=TemplatedParent}, Path=Propriete } Ou : {TemplateBinding Propriete}	À l'intérieur d'un template de contrôle : liaison sur la propriété «Propriete » de l'objet auquel est appliqué le template
{Binding RelativeSource={RelativeSource=PreviousData}, Path=Propriete }	Dans le cas d'une liaison sur une liste de données : liaison sur l'élément précédant l'élément courant. Utile par exemple pour calculer la distance de l'élément courant par rapport au précédent

### Mode de liaison :

Le mode de liaison définit dans quel sens est faite la liaison entre l'objet source et l'objet cible. Il est utilisé de la façon suivante : {Binding ..., Mode=XXX} avec XXX prenant comme valeur :

<b>TwoWay</b>	<b>La source et la cible reste synchronisées.</b> La source met à jour la cible lorsqu'elle change et inversement.
<b>OneWay</b>	Seules les <b>modifications sur la source sont répercutées sur la cible.</b> Les modifications de la cible ne changent pas les valeurs de la source.
<b>OneWayToSource</b>	C'est le fonctionnement inverse de la valeur précédente. Seules les <b>modifications sur la cible sont répercutées sur la source.</b>
<b>OneTime</b>	Les valeurs de la liaison ne sont fournies que de la source vers la cible et cela uniquement <b>lorsque l'application démarre ou à chaque changement du DataContext.</b>
<b>Default</b>	Le mode de liaison est défini par l'objet sur lequel il est appliqué.

### Convertisseurs et formatage des données :

Le type de données de la source n'est pas toujours celui que l'on veut utiliser pour l'affichage. Dans d'autres cas, il correspond mais on souhaite formater la façon dont il est affiché. Voici les différentes options possibles :

<b>StringFormat={}{format}</b>	Liaison sur le DataContext courant en formatant la donnée (de type String) en utilisant le format « format » compatible avec la méthode String.format(). Les {} sont nécessaires car nous n'ajoutons pas directement du texte après le signe égal et le compilateur n'aime pas cela !
<b>TargetNullValue=valeur</b>	Lorsque la source prend la valeur <i>null</i> alors la valeur « valeur » est utilisée
<b>FallBackValue=valeur</b>	Lorsqu'une erreur se produit, la valeur « valeur » est utilisée
<b>Converter=convertir</b>	Permet de spécifier un convertisseur de type pour convertir une valeur de type A de l'objet source vers une valeur de type B pour l'objet cible. Par exemple : convertir un booléen vers une valeur de visibilité.
<b>ConverterParameter=parametre</b>	Permet de fournir un paramètre « parametre » au convertisseur. Cela ne peut pas être une liaison mais peut être une ressource (par exemple : {StaticResource uneResourceStatique})
<b>ConverterCulture=culture</b>	Indique au convertisseur qu'il doit utiliser la culture « culture »

## Événement déclencheur de la mise à jour de la liaison :

Les liaisons de données ne sont pas totalement synchrones et ce sont certains événements qui déclenchent la synchronisation. On paramètre l'événement de la façon suivante : `{Binding ..., UpdateSourceTrigger=XXX}` avec XXX prenant comme valeur :

<b>PropertyChanged</b>	La mise à jour s'effectue immédiatement à chaque changement de valeur
<b>LostFocus</b>	La mise à jour s'effectue lorsque le contrôle où est faite la liaison perd le focus
<b>Explicit</b>	La mise à jour doit être faite explicitement sur l'objet <code>BindingSource</code> correspond à la liaison via la méthode <code>UpdateTarget</code> . Voici un exemple : <pre>BindingExpression be = txtBx.GetBindingExpression(TextBox.TextProperty); be.UpdateTarget();</pre>
<b>Default</b>	La cible de la liaison décide quand elle se met à jour

## Paramétrage de la liaison

<b>IsAsync</b>	<b>La liaison est faite de manière asynchrone si cette valeur est fixée à True.</b> Cela permet <b>d'éviter un blocage de l'affichage</b> lorsque la valeur est longue à calculer. Pendant le calcul, c'est la valeur <code>FallBackValue</code> qui est utilisée.
<b>BindsDirectlyToSource</b>	Lorsque l'on se lie à un fournisseur de données ( <code>DataProvider</code> comme par exemple un <code>ObjectDataProvider</code> ), fixer cette valeur à <code>False</code> indique que la source de données est le fournisseur de données. Sinon la liaison est faite directement sur la source de données du fournisseur.
<b>NotifyOnTargetUpdated</b>	Lorsque cette propriété est fixée à <code>True</code> , cela permet de lever un événement lorsque la cible est mise à jour. Le gestionnaire d'événement utilisé est défini par la propriété <code>TargetUpdated</code> sur l'élément à surveiller.
<b>NotifyOnSourceUpdated</b>	Lorsque cette propriété est fixée à <code>True</code> , cela permet de lever un événement lorsque la source est mise à jour. Le gestionnaire d'événement utilisé est défini par la propriété <code>SourceUpdated</code> sur l'élément à surveiller.

## Règles de validation des données de la liaison

Lorsque l'utilisateur entre des données dans un contrôle ayant une liaison de données, celles-ci peuvent être validées via certaines règles. Les différentes erreurs sont accessibles via la propriété `Validation.Errors` de l'élément cible.

<b>ValidationRules</b>	Permet d'indiquer une liste de règles de validation lorsque l'utilisateur entre des données. Une règle de validation contient une méthode <code>Validate()</code> qui retourne un objet de type <code>ValidationResult</code> qui indique si la donnée fournie est valide ou non.
<b>BindingGroupName</b>	Permet d'indiquer un nom pour former des groupes de validation afin de valider des soumissions de données par groupe.
<b>ValidatesOnExceptions</b>	En fixant à <code>True</code> cette propriété, les exceptions levées par la mise à jour de la source sont considérées comme des erreurs de validation.
<b>ValidatesOnDataErrors</b>	En fixant à <code>True</code> cette propriété, la liaison utilise l'interface <code>IDataErrorInfo</code> de l'objet source afin d'effectuer la validation
<b>UpdateSourceExceptionFilter</b>	Permet de définir un gestionnaire d'événements qui s'abonnera aux événements levés par le moteur de liaison durant la mise à jour de la source. Cette propriété ne peut être fixée que lorsqu'une <code>ExceptionValidationRule</code> est ajoutée aux règles de validation
<b>NotifyOnValidationError</b>	Si cette propriété est fixée à <code>True</code> alors il est possible de s'abonner à l'événement <code>Validation.Error</code> sur l'élément où est faite la liaison

