TListView, apparence dynamique et image

Par Serge Girard

Date de publication : 5 mai 2019

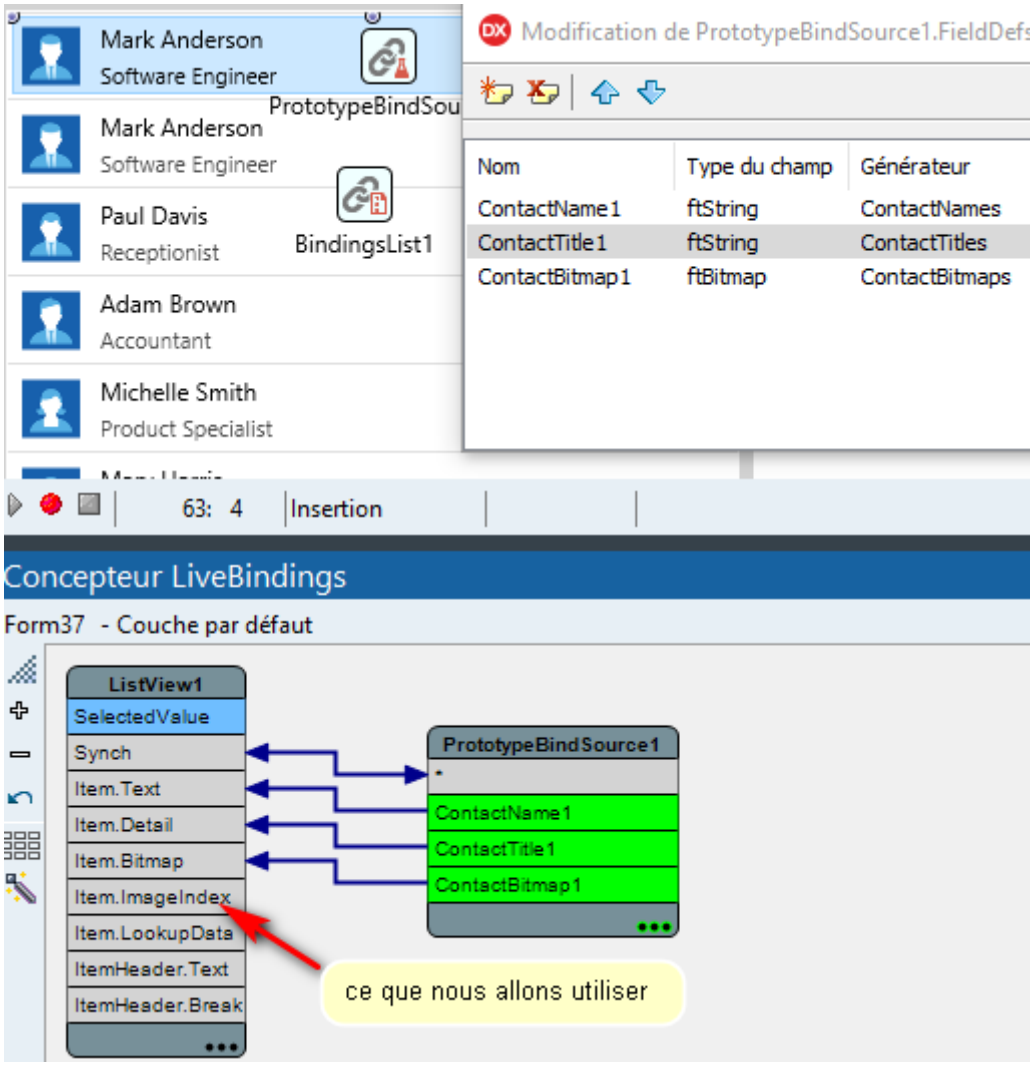
L'objectif est d'afficher une d'icône selon l'état d'une donnée. Il existe plusieurs apparences prédéterminées qui permettent cela en fait toutes celles dont le nom commence par ***Image***. Le mode d'apparence dynamique n'offre pas cette possibilité. Le but de ce petit article est de montrer comment réaliser la même chose avec un peu de code.

En complément sur Developpez.com
- Mettre de la couleur dans un TListView
- Personnaliser un TListView : ajouter des pieds de groupes
- LiveBindings de A à … : TPrototypeBindSource

I - Apparence prédéterminée

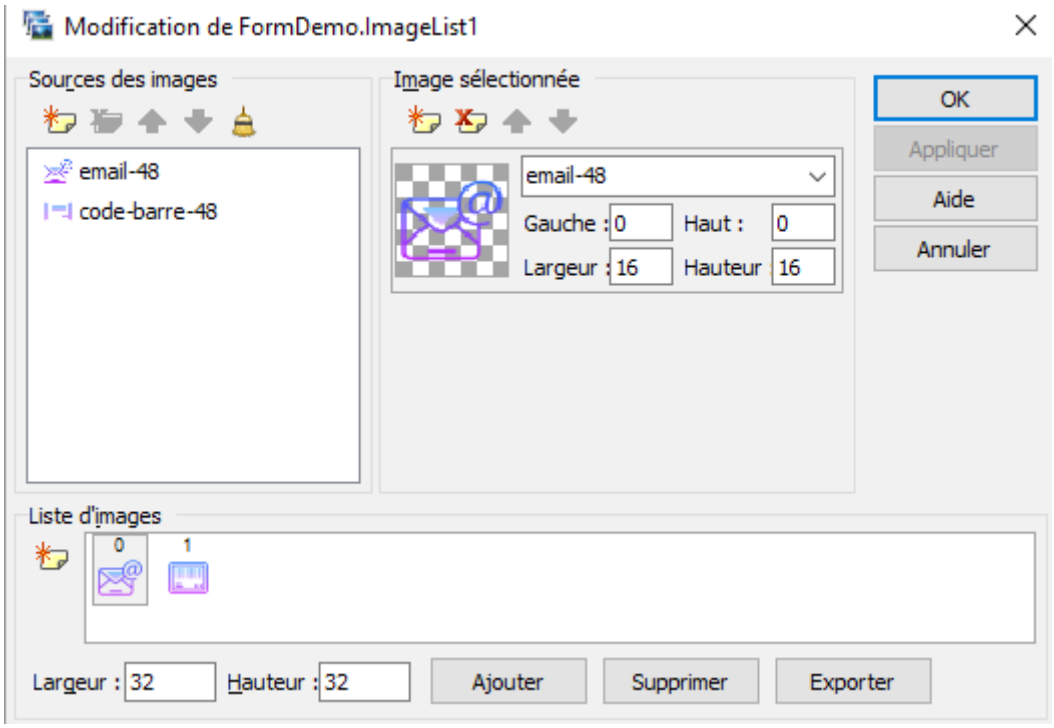
Avant de passer à l'apparence dynamique passons d'abord par une apparence prédéterminée pour voir comment cela se réalise.

En préambule il me faut peut-être éclaircir un premier point. Il est très facile de créer une liste qui affiche des images pour peu d'utiliser la bonne apparence d'élément de liste et ce sans une once de codification. Comme exemple cette simple petite réalisation au moment du design le prouve.



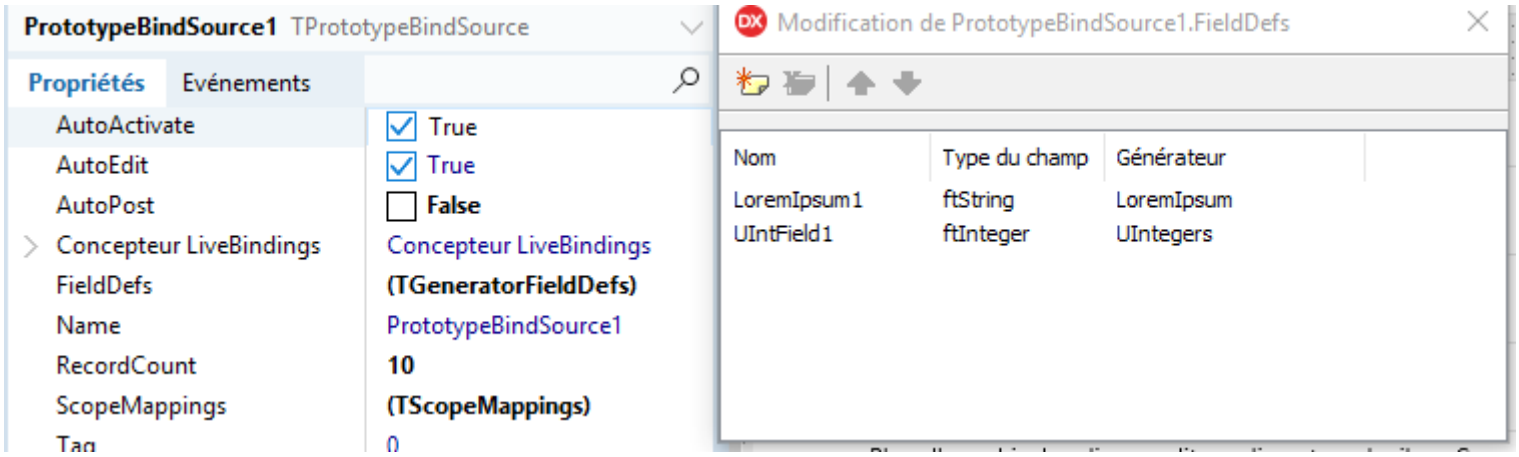
Cependant, dans cette première image, il s'agit de l'affichage d'une image contenue dans la source des données. Peut-être moins connue est l'utilisation de la propriété de l'élément de liste nommé ImageIndex or c'est cette propriété, associée à une liste d'images, que nous allons utiliser.

1 Sur une forme FMX ajoutez un **TImageList** que vous remplirez avec les images désirées.



Bien évidemment il nous faut une source de données. Pour les besoins de la démonstration je vais utiliser un **TProtypeBindSource** qui contiendra un champ texte et un champ numérique.

2	Déposez sur la forme un TProtypeBindSource Ajoutez-y un champ texte (*LoremIpsum*) et un champ numérique entier (*UIntegers*)



Gardez en mémoire qu'il s'agit d'une génération aléatoire, le champ numérique sera donc compris entre 0 et 200.

3	Déposez un **TListView**. Modifiez l'apparence (***ItemAppearance/ ItemAppearance***) de façon à ce que l'élément de liste contienne une image. Vous avez le choix entre plusieurs possibilités :

	ImageListItem; ***ImageListItemBottomDetail;*** ***ImageListItemBottomDetailRightButton;*** ***ImageListItemRightButton***. Reliez le **TImageList** au **TListview** grâce à la propriété ***ImageList*** de ce composant.

Passons maintenant à la partie liaison (**LiveBindings**) entre la liste et les données.

Pour mémoire, il y a plusieurs possibilités pour appeler le concepteur visuel de liaisons

En utilisant le menu contextuel du composant (clic droit sur le composant)	
En utilisant le menu situé en bas de l'inspecteur d'objet	
En utilisant le menu de l'EDI	Voir Fenêtre d'outils Concepteur LiveBindings

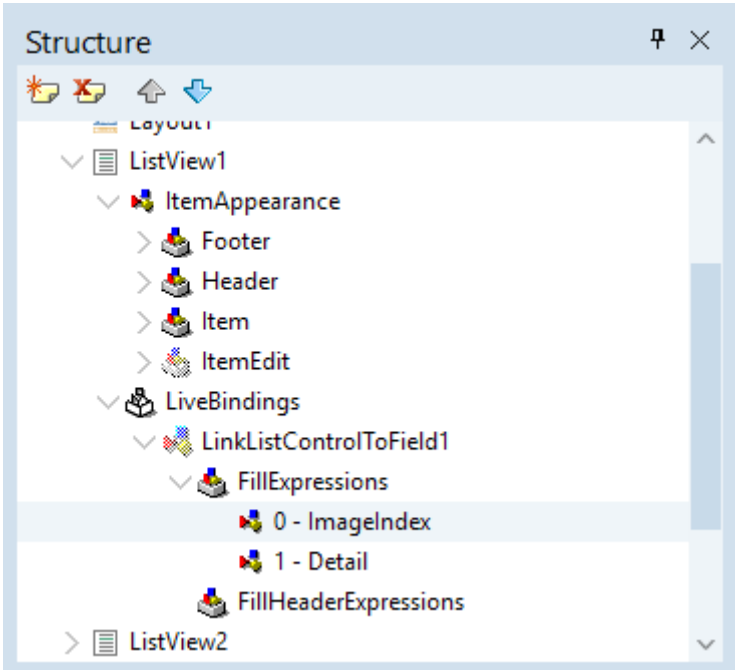
*Toutes les versions de Delphi ne proposent pas ce concepteur néanmoins c'est le cas partir de la version Tokyo 10.2**

** hormis sa version Starter*

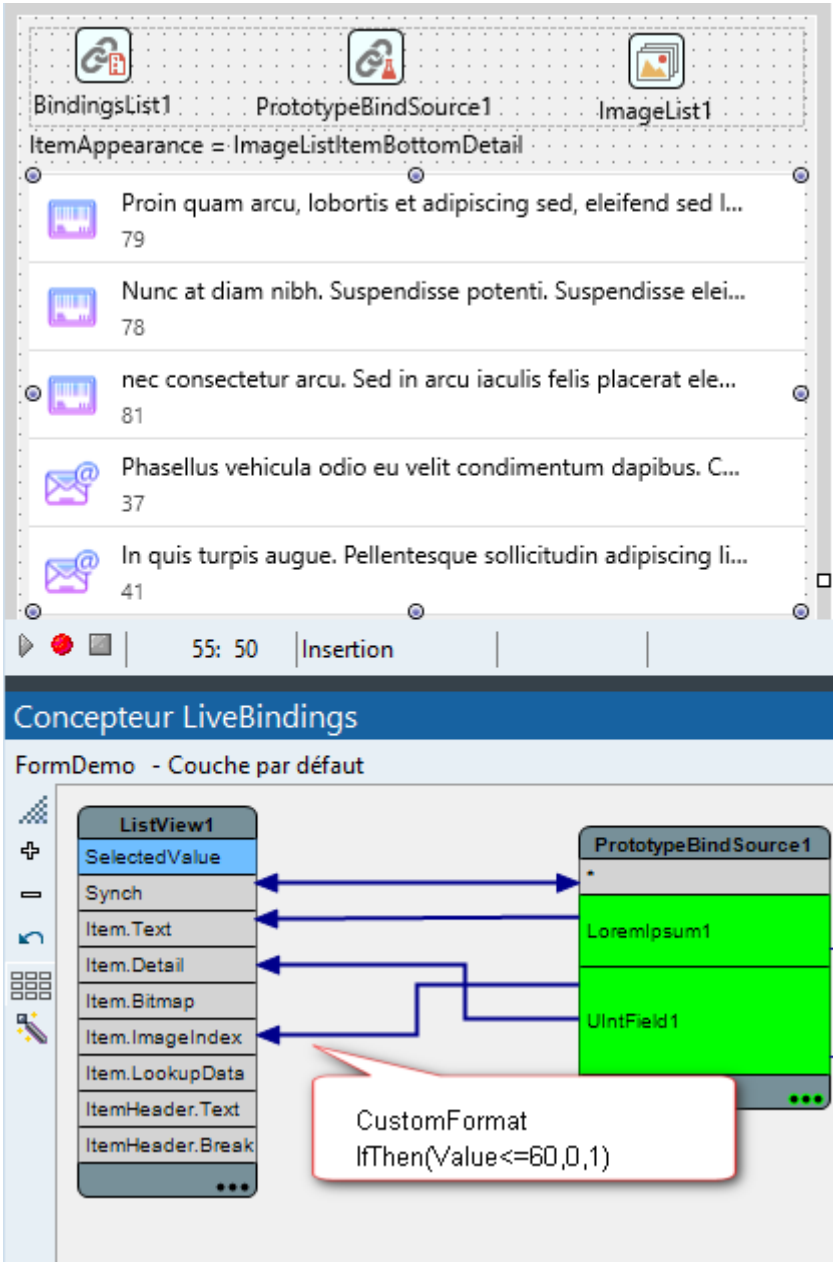
4	Liez la source de données (Synch ↔ *) Liez le champ texte (LoremIpsum1 ↔ Item.Text) Liez le champ numérique (UIntField1 ↔ Item.Detail) * *optionnel* Liez le champ numérique (UIntField1 ↔ Item.ImageIndex)

Bien évidemment comme il n'y a pas 200 images dans notre **TImageList**, il y a peu de chance que des icônes soient affichés. Pour pallier j'ai décidé que toute valeur inférieure ou égale à 60 afficherai l'icône 0 de ma liste et que sinon ce serait l'icône 1.

5	Dans le panneau de structure sélectionnez l'expression *ImageIndex* et indiquez dans la propriété CustomFormat la formule IfThen(value<=60,0,1)



Et voilà, pas besoin d'exécuter le programme, le résultat est déjà visible en mode design !

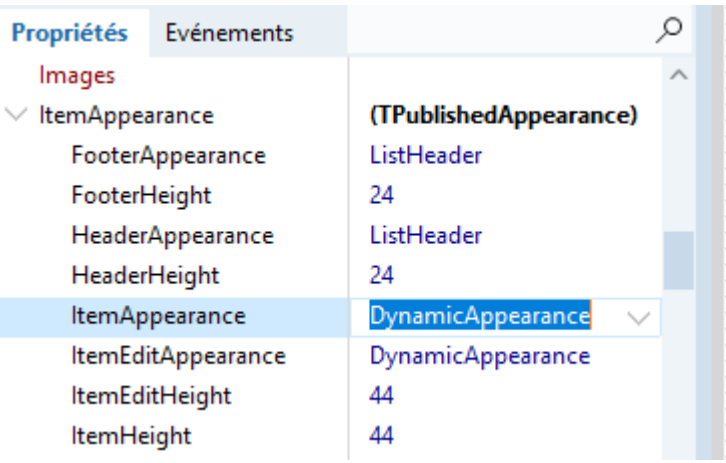


II - Apparence dynamique

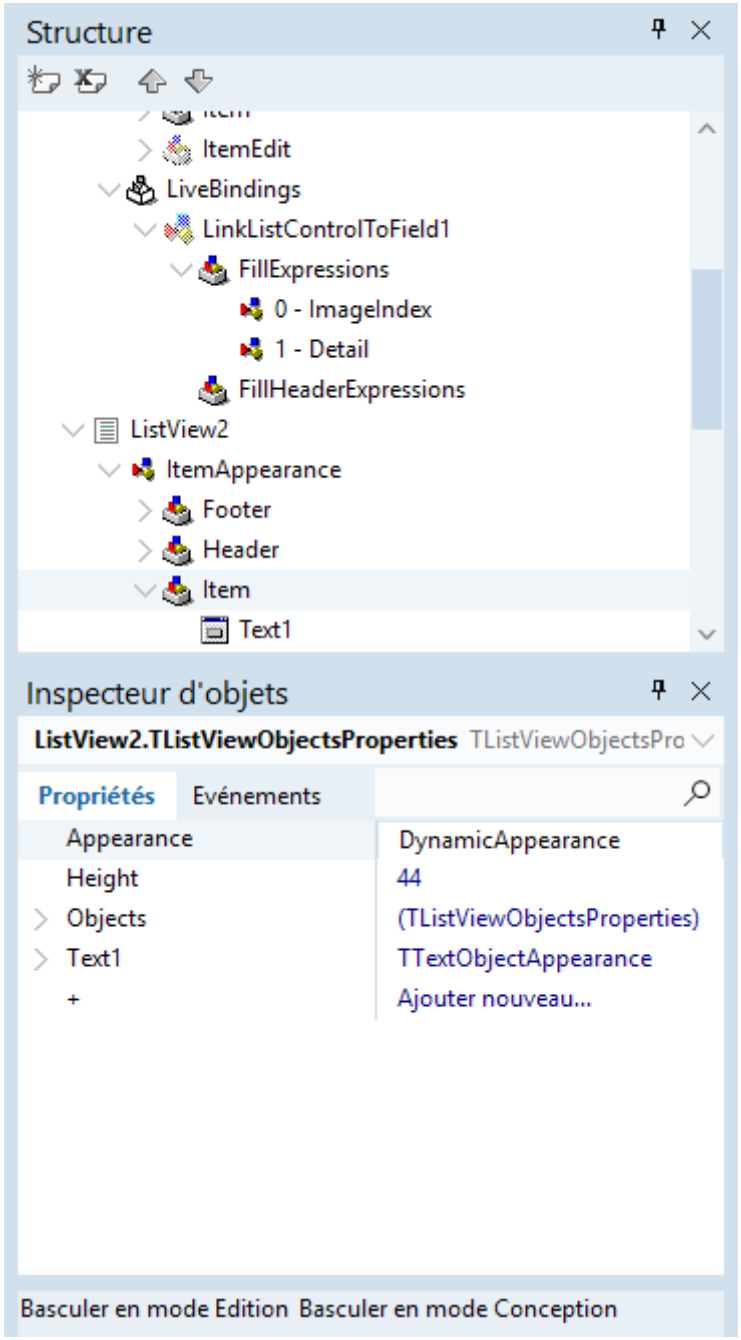
Si les apparences prédéterminées couvrent la plupart des besoins, nous sommes restreints à n'afficher que deux éléments (sauf à utiliser des concaténations). Depuis Delphi XE7 il est possible de personnaliser l'apparence de vue d'une liste (cf. **dockwiki**), permettant déjà de modifier les apparences prédéterminées mais, surtout, introduisant le mode dynamique.

II-A - Mise en place

1	Ajoutez un nouveau **TListView**. Sélectionnez une apparence dynamique pour vos éléments de liste (**DynamicAppearance**)



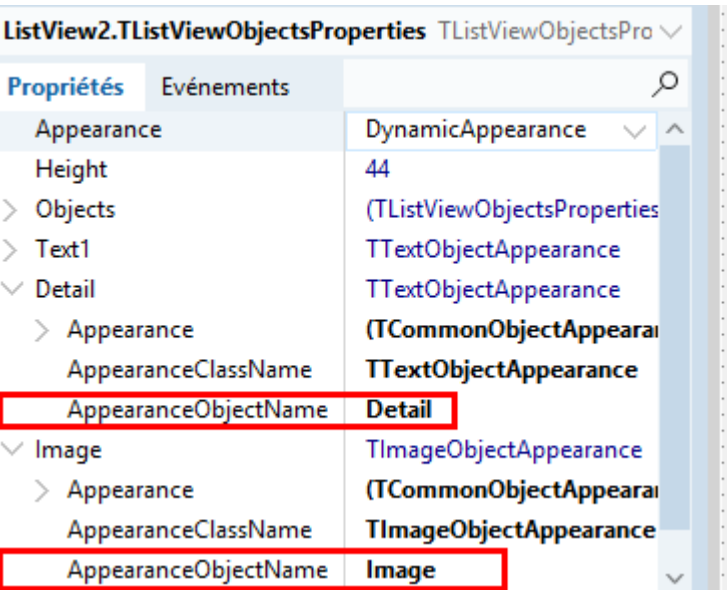
2	Sélectionnez ***Item*** de ***ItemAppearance*** dans la vue structure. Puis, dans l'inspecteur d'objets, cliquez sur la propriété + et ajoutez successivement un objet texte (***TTextObjectAppearance***) et un objet image (***TImageObjectAppearance***)

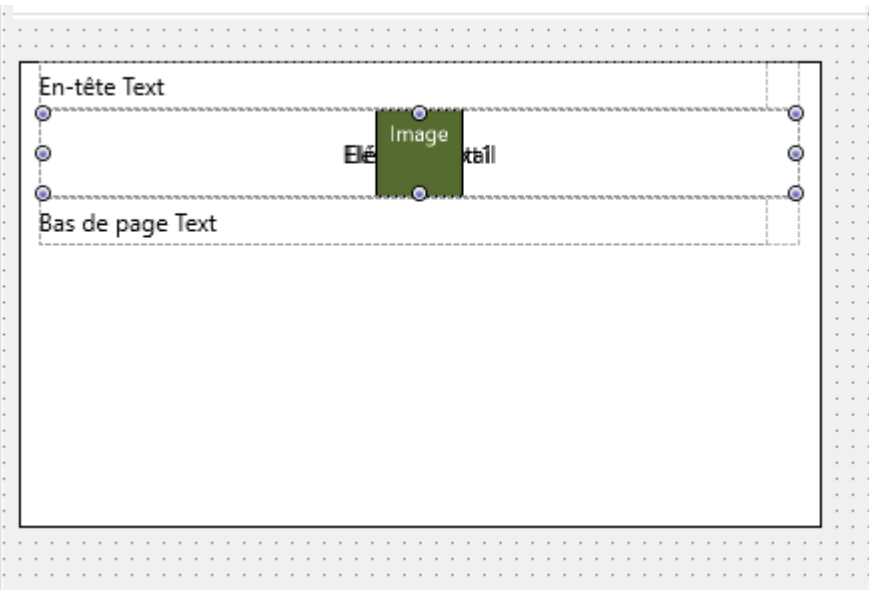


3	Basculez en mode conception soit par le menu situé en bas de l'inspecteur d'objet, soit en utilisant le menu contextuel du composant.

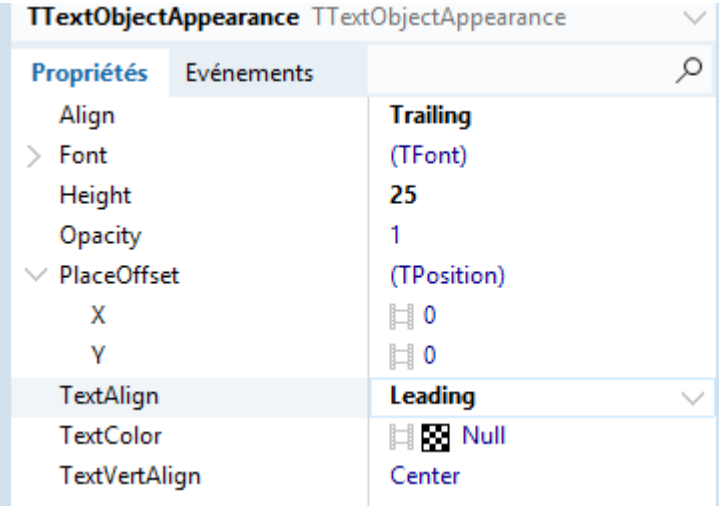
N'hésitez pas à changer le nom de l'objet ajouté

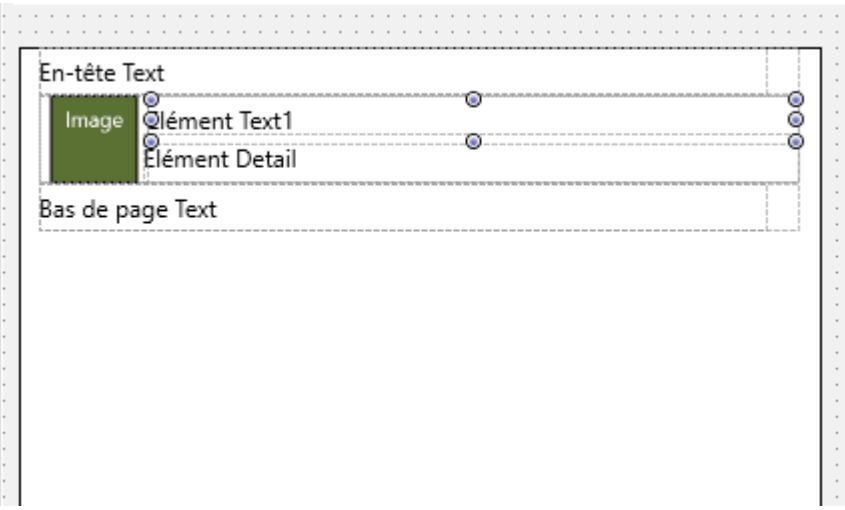
*(propriété **AppearanceObjectName**)*



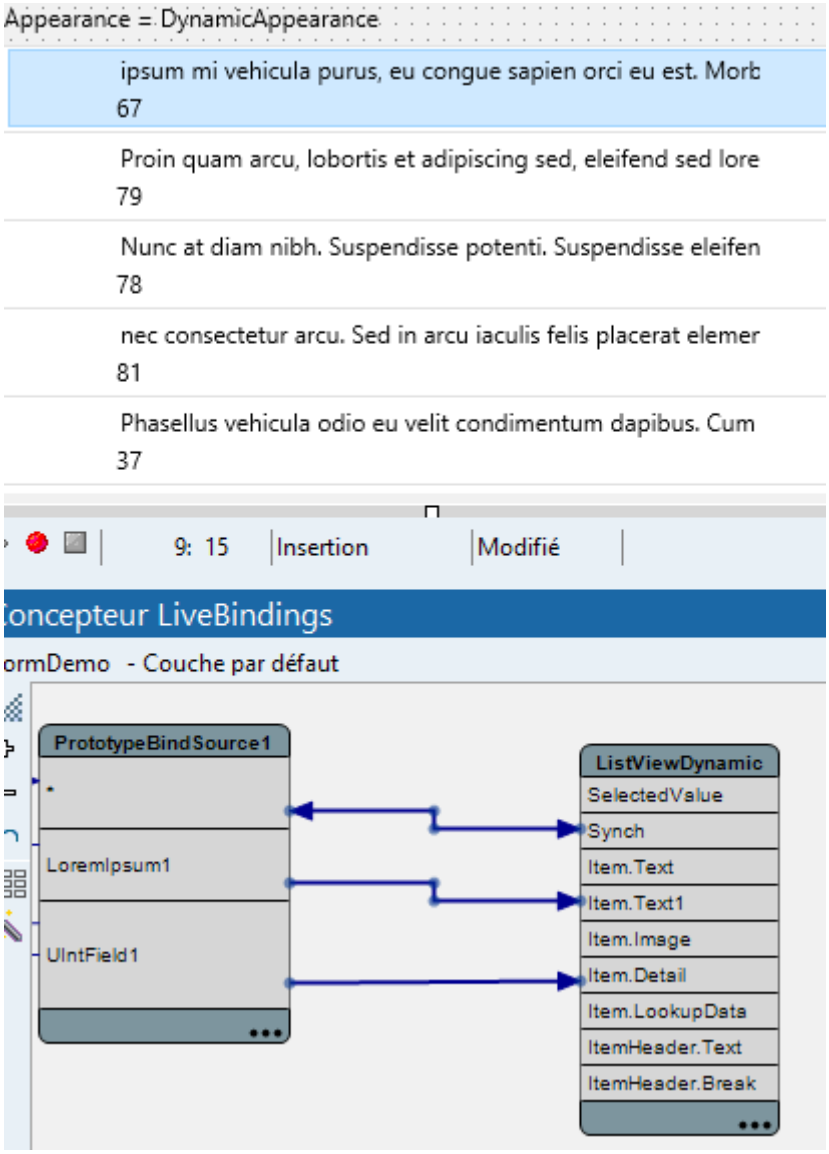


4	Arrangez la présentation comme bon vous semble.





5	Désactivez le mode conception. Passer sur le concepteur de liaisons (**Livebindings**) Liez la source de données (Synch ↔ *) Liez le champ texte (LoremIpsum1 ↔ Item.Text1) Liez le champ numérique (UIntField1 ↔ Item.Detail) * *optionnel*



Que constate-t-on ? Premier « piège » **Item.Text**, bien que disponible, ne sera pas affiché. Il faut utiliser **Item.Text1**.

J'ai mis piège entre guillemets car cette particularité s'avère pratique puisque, ne s'affichant pas, il y a là moyen d'y stocker une valeur (j'aurais pu y lier le champ).

Mais le problème principal, déclencheur de ce tutoriel, c'est qu'il n'y a pas l'élément **Item.ImageIndex** qui nous était proposé chapitre I.

II-B - Solution

Si, lors du chapitre I, aucun code n'était nécessaire, c'est bien en codant que nous arriverons à résoudre l'objectif. Quel évènement sera à utiliser ? **OnUpdateObjects** du **TListView** sera le candidat idéal.

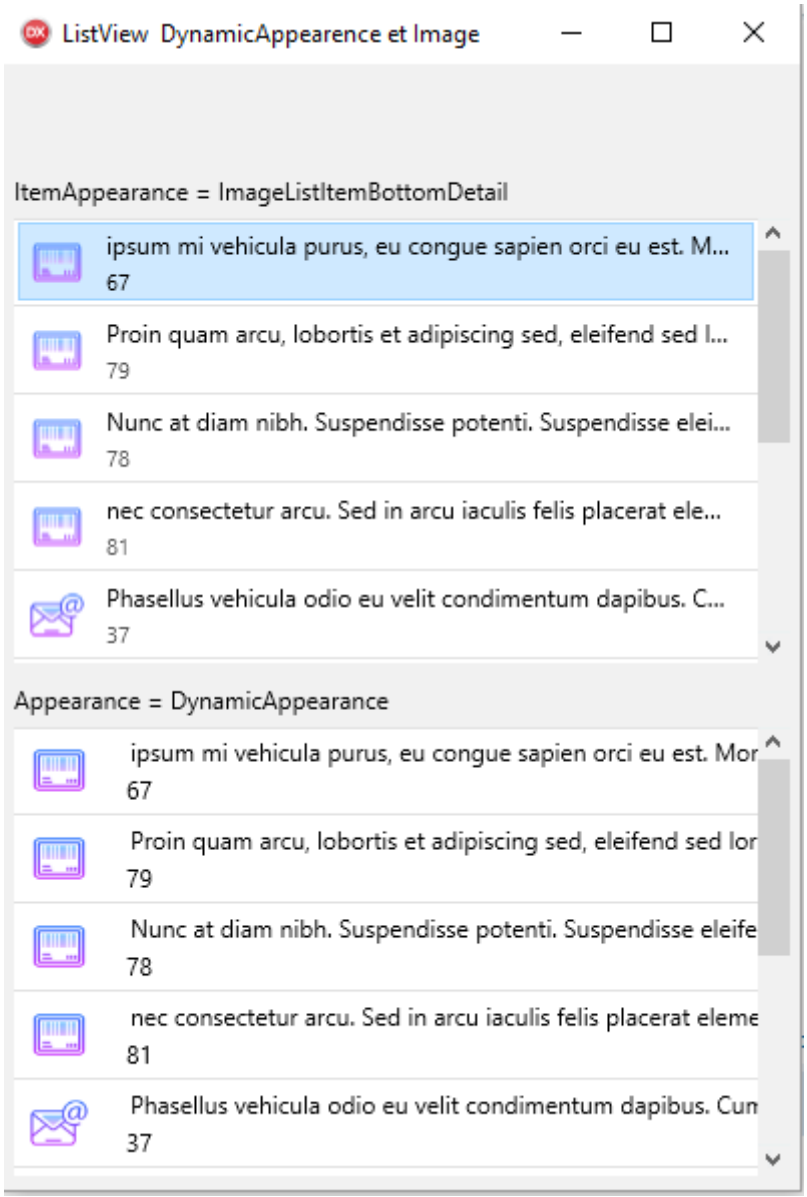
Pourquoi OnUpdateObjects et non OnUpdatingObjects ?

Que nous faut-il#? Accéder à l'objet nommé**Image** mais aussi à la valeur liée à l'objet **Detail**.

L'accès à ces deux objets peut se faire en utilisant la fonction **FindObject** (considérée obsolète et remplacée par **FindDrawable**) ou **FindObjectT<T>** (<T> indiquant le type d'objet à obtenir).

À partir du moment où ces deux éléments sont trouvés il suffira de coder ce qu'a fait l'expression du **CustomFormat** du chapitre I.

```
procedure TFormDemo.ListViewDynamicUpdateObjects(const Sender: TObject;
  const AItem: TListViewItem);
var AListItemBitmap : TListItemImage;
    AListItemText : TListItemText;
    i : Integer;
begin
AListItemBitmap:=AItem.Objects.FindObjectT<TListItemImage>('Image');
AListItemText:=AItem.Objects.FindObjectT<TListItemText>('Detail');
if Assigned(AListItemBitmap) AND Assigned(AListItemText) then
 begin
     i:=StrToIntDef(AListItemText.Text,-1);
     if i<=60 then  AListItemBitmap.ImageIndex:=0
              else  AListItemBitmap.ImageIndex:=1;
 end;
end;
```



III - Pour finir

Après avoir révisé ou découvert l'utilisation d'une liste d'images en association à un TListView, nous avons découvert que l'apparence dynamique d'un élément de liste n'offrait pas cette possibilité à moins de passer par un peu de code.

Je tiens à remercier l'équipe rédactionnelle, pour la relecture technique et pour les corrections grammaticales et orthographiques.