
# AJUI_Breadcrumb

# Manuel d'utilisation

| Version Control | Date | Commentaire (Changement) | Auteur |
|---|---|---|---|
| **1.0** | 04.11.2019 | Première version | Gary Criblez |
| **1.0.1** | 08.11.2019 | Arrondies d'angle de section | Gary Criblez |
| **1.1** | 09.12.2019 | Utilisation du placeholder "#" avec la propriété PicturePath dans une section | Gary Criblez |
| **1.2** | 17.01.2020 | Ajout du modèle "groupedButtons" | Gary Criblez |
| | | | |

# 1 Introduction

## 1.1 Qu'est-ce que AJUI_Breadcrumb ?

AJUI_Breadcrumb est un composant développé dans le langage 4D. Son utilisation est destinée aux développeurs 4D. Il permet de générer et afficher dynamiquement des Breadcrumbs dans le contexte d'un formulaire.

**Qu'est-ce qu'un Breadcrumb ?**

Le Breadcrumb (ou fil d'Ariane) désigne un élément de navigation principalement utilisé sur le web. Il est utilisé pour indiquer à l'utilisateur à quel niveau hiérarchique ce dernier se situe.

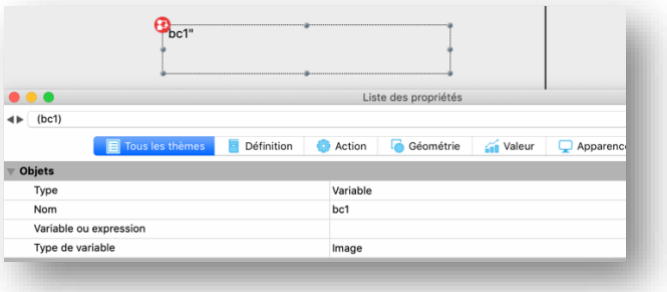
Dans le cadre d'une application métier réalisée avec 4D, le Breadcrump permettra de visualiser l'ensemble des étapes d'un processus, de savoir l'étape en cours et de revenir sur une des étapes précédentes d'un processus de traitement ou de saisie.

Pour ce faire le Breadcrumb est composé de sections qui vont permettre de naviguer entre différentes pages d'un formulaire ou d'afficher des sous formulaires spécifiques suivant la hiérarchie de la logique métier de l'application.

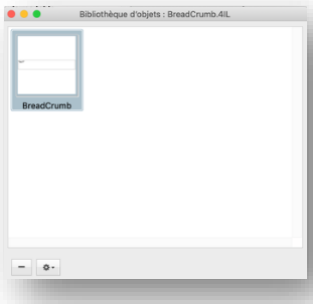
Le composant **AJUI_Breadcrumb** est défini à l'aide d'une variable objet représentant une instance d'AJUI_BreadCrumb. Cette variable contient un ensemble de propriétés et formules (fonctions membres) permettant de définir la structure et le contenu du Breadcrumb et de le générer graphiquement dans un formulaire 4D.

## 1.2 Composition d'un Breadcrumb

Fondamentalement, le Breadcrumb généré par AJUI_BreadCrumb est une variable objet de formulaire de type image que vous pouvez créer vous-même dans le contexte d'un formulaire.




Néanmoins, nous vous mettons à dispositions un modèle prédéfini de cette variable objet de formulaire contenant toutes les propriétés appropriées pour la création d'un BreadCrumb. Celle-ci est fournie dans un fichier de librairie d'objet « BreadCrumb.4dlibrary ».




## 1.3 Structure et modèle de Breadcrumb

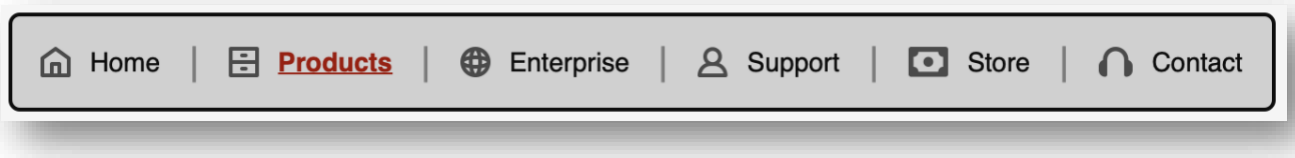
Un Breadcrumb généré par le composant est composé d'un "*conteneur*" qui peut être dimensionné manuellement ou dynamiquement. Le conteneur va contenir des sections. Vous pouvez définir le nombre de sections que vous souhaitez.

Une section est associée à un "modèle" et elle peut contenir des éléments textuels, une image et un callback. Elle possède également un identifiant (id) la rendant unique et permettant de l'identifier lors de l'interaction avec le Breadcrumb.

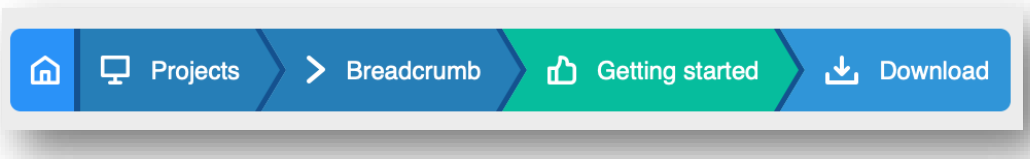
AJUI_Breadcrumb offre deux principaux modèles de Breadcrumb.

- Le modèle **simple**
- Le modèle **arrow**
- Le modèle **groupedButtons**

Le modèle **simple** est un enchaînement de *sections*, dans un seul *conteneur*, séparées par un *divider* (séparateur) de type texte, c'est-à-dire une chaîne de caractère quelconque (par exemple : ">", "/", "|", etc.)

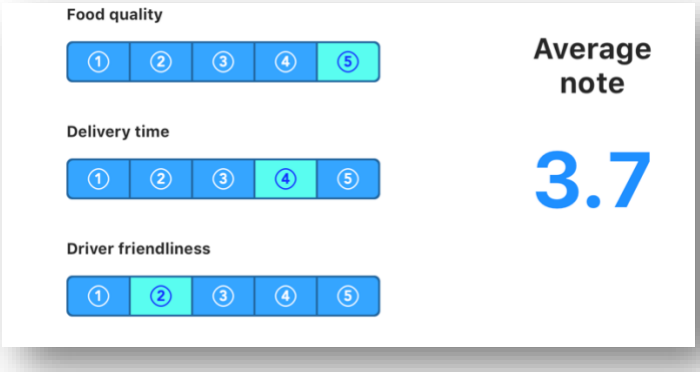



Le modèle **arrow** est différent. Chaque section possède son *conteneur* dont les extrémités avant et arrière peuvent être représentées par une forme de flèche ou un trait vertical. Un *divider* (séparateur) permet de définir la taille du trait de chaque flèche.



Le modèle **groupedButtons** est un peu particulier parce que ce n'est pas exactement un nouveau modèle. Il se base sur le modèle **arrow** en utilisant des propriétés prédéfinies afin de garder un design propre à ce modèle.

Lors de l'utilisation de ce modèle, les sections peuvent être soit type « standard », soit de type « current ». Le type « current » désigne la section sélectionnée et il n'y en peut y avoir qu'une seule à la fois. En clair, le modèle fonctionne comme des boutons radios.




⚠ *Petite précision sur les identifiants (id) de sections. Dans le cadre du modèle **simple**, l'id permet l'interaction de la souris avec le texte et l'image de la section. Concernant le modèle **arrow et groupedButtons**, l'id s'applique sur toute la zone graphique du conteneur de la section.*

### 1.3.1 Section - types et états

Il existe cinq types de section :

- **Standard**
- **First**
- **Current**
- **Previous**
- **Next**

A chaque type de section sont associés quatre états :

- **default**
  C'est l'état d'une section lorsqu'un utilisateur n'interagit pas avec elle et que celle-ci n'est pas désactivée.
- **hover**
  Cet état intervient lorsque l'événement « On Mouse Enter » ou « On Mouse Move » (id de section trouvé) se déclenche. Concrètement, cela correspond au survol de la section avec la souris. Il se termine sur l'événement « On Mouse Leave » ou sur « On Mouse Move » (id de section différent ou nul).
- **active**
  Cet état intervient sur l'évènement « On Clicked » et se termine sur l'événement « On Mouse Up. À noter qu'il a la priorité sur l'état « hover ». Concrètement, l'utilisateur clique sur une section et l'état perdure jusqu'au relâchement de celui-ci.
- **disable**
  C'est l'état que prend une section si sa propriété « disable » est à vrai. Lorsque la section est « disable », seul cet état est pris en compte à l'exclusion de tous les autres (hover et active).

Une section peut prendre une apparence et afficher un contenu différent en fonction de son type et de son état.

Les états différents de l'état « default », héritent de l'ensemble des propriétés de celui-ci. Chaque propriété peut se voir affecter des exceptions.

### 1.3.2 Les événements et callbacks

Le Breadcrumb nécessite que plusieurs événements soient activés sur l'objet de formulaire image afin de pouvoir gérer les différents états. Voici la liste de ceux-ci :

- On Load
- On Clicked
- On Double Clicked (optionnel)
- On Mouse Enter
- On Mouse Leave
- On Mouse Up
- On Mouse Move

Pour les deux événements (On Clicked, On Double Clicked) il est possible de leur associer des callbacks à l'aide des fonctions membres.

Attention : Toutes méthodes utilisées comme callback doivent être partagées (propriétés de la méthode) avec le composant pour que celui-ci puisse l'appeler. Pour éviter qu'une erreur se produise pour ce genre de cas, le composant vérifiera si la méthode est bien partagée et si ce n'est pas le cas, il la partagera de lui-même lors de l'exécution d'un callback. De plus, le composant proposera de créer la méthode de callback si elle n'existe pas lors de l'utilisation des **Setters** pour les callbacks et également avant l'exécution d'un callback.

### 1.3.3 Gestion des exceptions et constantes

Le composant utilise un système d'exception afin de distinguer quelle valeur des propriétés liées aux états il doit utiliser lors de la génération du Breadcrumb.

Il faut savoir que lors de la création d'une instance, le composant affecte des valeurs par défaut aux propriétés de type « **standard** » et pour l'état « **default** ». Donc, quand vous allez utiliser des fonctions membres liées aux états (voir le sous-chapitre 3.9 : Section type), vous devrez passer en premier paramètre, une constante qui désignera le type et l'état pour lequel vous voulez recevoir ou passer une valeur (Getter/Setter).

Exemple :

```
  // Setter
$bc.SectionBgColor(AJUI_bc_standard_default;$color)
  //Getter
$colorSection:=$bc.SectionBgColor(AJUI_bc_standard_default)
```

**Comment le composant gère les exceptions :**

Pour expliquer cela, nous allons prendre un exemple concret. J'ai une section de type « first » et l'on veut définir son état « hover ».

Le composant va donc d'abord vérifier s'il existe des exceptions au niveau de l'objet « *monInstance.breadCrumb.sectionTypes.first.hover* ». S'il en trouve, il utilise ces valeurs. S'il ne trouve pas d'exceptions pour l'ensemble ou pour une partie des propriétés, il va contrôler les exceptions pour « *monInstance.breadCrumb.sectionTypes.first.default* ». Après cela, il vérifiera « *monInstance.breadCrumb.sectionTypes.standard.hover* » et pour finir « *monInstance.breadCrumb.sectionTypes.standard.default* » s'il reste des propriétés à définir.

Il n'ira pas plus loin, car les valeurs de « standard-default » ne sont pas des exceptions, mais les valeurs de bases.

| | First | First | First | Standard | Standard |
|---|---|---|---|---|---|
| | hover (final values) | hover | default | hover | default |
| bgColor | #CECECE | | #CECECE | | #D4E3FE |
| borderColor | Red | Red | | | Black |
| borderSize | 2 | | | 2 | 0 |
| fontColor | White | White | Blue | | #0148AA |
| fontName | Arial | | | | Arial |
| fontSize | 12 | | | | 12 |
| fontStyle | bold | bold | | | None |
| pictureOpacity | 80 | 80 | | | 100 |
| replacingColor | none | | | | none |

**Liste des constantes :**

- *AJUI_bc_standard_default*
- *AJUI_bc_standard_hover*
- *AJUI_bc_standard_active*
- *AJUI_bc_standard_disable*

- *AJUI_bc_current_default*
- *AJUI_bc_current_hover*
- *AJUI_bc_current_active*
- *AJUI_bc_current_disable*

- *AJUI_bc_first_default*
- *AJUI_bc_first_hover*
- *AJUI_bc_first_active*
- *AJUI_bc_first_disable*

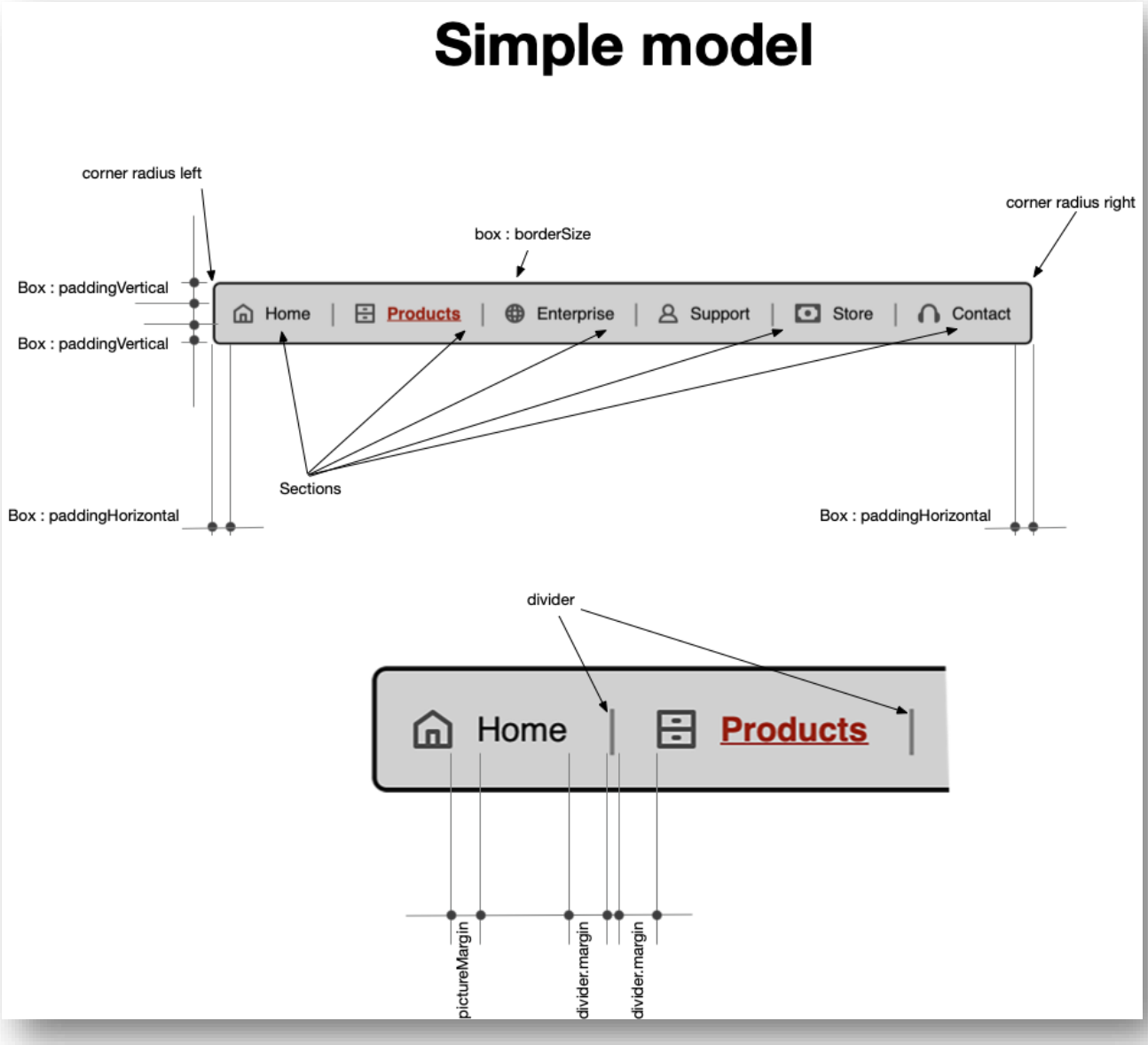
- *AJUI_bc_previous_default*
- *AJUI_bc_previous_hover*
- *AJUI_bc_previous_active*
- *AJUI_bc_previous_disable*

## 1.4 Aperçu technique

Dans cette partie, nous vous mettons à disposition deux schémas décrivant comment les modèles sont structurés au niveau de la construction graphique du conteneur et des sections.

### 1.4.1 Breadcrumb simple

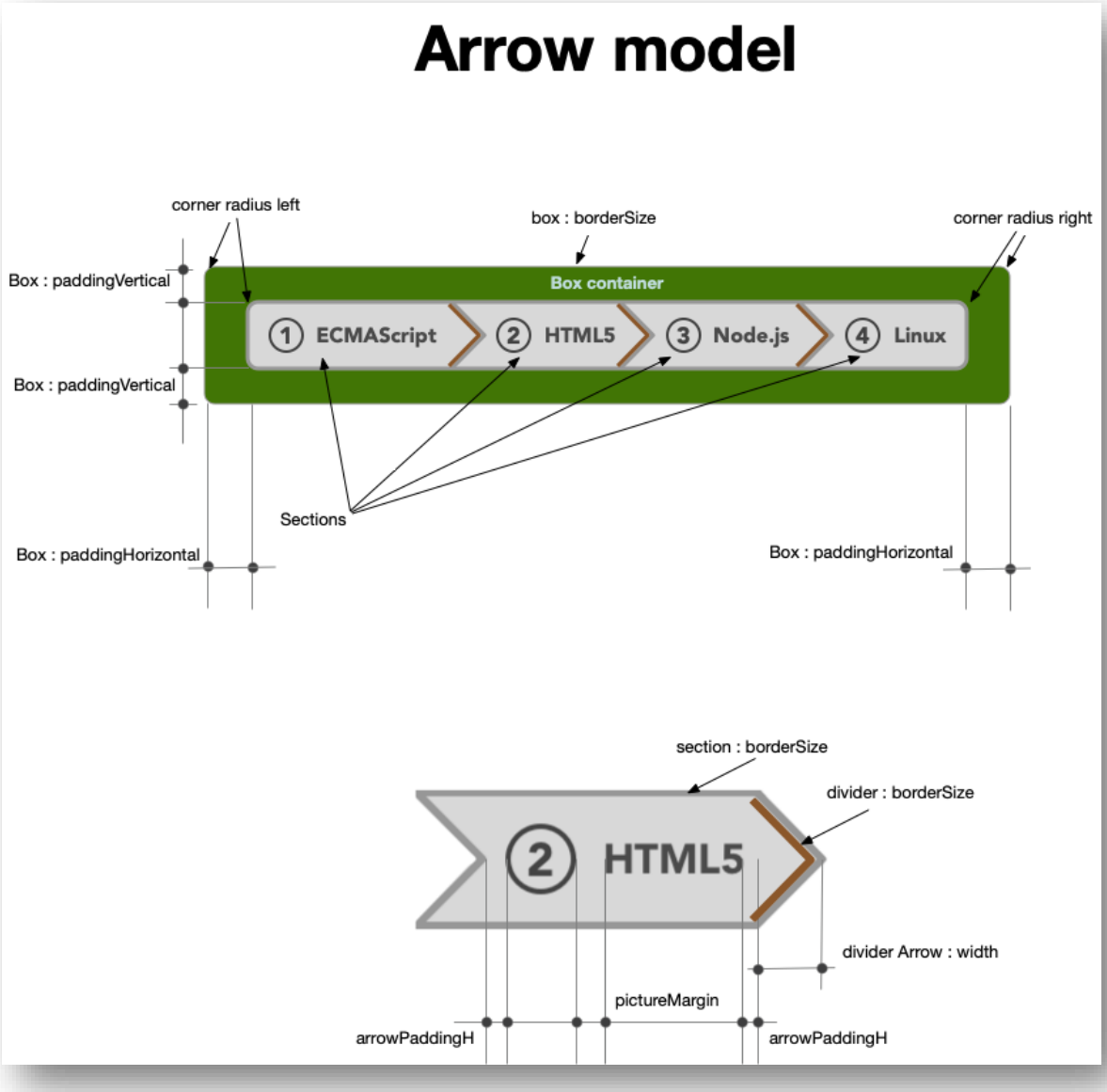
Le modèle « simple » est un enchaînement de sections contenant du texte et/ou une image séparée par un divider et une marge qui s'applique sur chacun des côtés du divider.



### 1.4.2 Breadcrumb arrow

Le modèle « arrow » est un enchaînement de sections dont la partie droite est terminée par une pointe (flèche) et dont on peut définir la taille. La partie arrière de la section reproduit la forme de la section précédente. Il est possible de définir un divider (trait) qui viendra se plaquer à l'intérieur de la partie avant de la section et reproduira la forme de la flèche. Il est possible de séparer les sections via une marge.

On peut définir une taille de bordure aux sections qui sera placée à l'intérieur de celles-ci. La première et la dernière section se voit appliquer les valeurs définies des arrondis d'angle (corner radius) du container, en leur appliquant un facteur proportionnel égal à la différence de hauteur entre les sections et le container, ceci afin de maintenir et garantir un parfait parallélisme entre section et container. La première et/ou la dernière section peut être définies avec ou sans pointe (flèche).



### 1.4.3 Breadcrumb groupedButtons

Le modèle « groupedButtons » possède la même structure que le modèle « arrow ». Il diffère de ce dernier par son comportement lors d'un clic sur une section. Ainsi lors d'un clic il change son état à "current" et définit les autres sections à l'état "standard. Il se comporte ainsi comme un bouton radio. Par ailleurs, plusieurs propriétés sont prédéfinies afin de conserver une logique dans le design de ce modèle (par exemple, les sections ne se terminent pas en forme de flèche).

**Propriétés et valeurs prédéfinies** :

- dynamicSize (box) : True
- bgColor (box) : valeur de la couleur de la bordure de la box
- margin (divider) : valeur de la taille de la bordure de la box
- padding horizontal (box) : 0
- padding vertical (box) : 0
- width (arrow) : 0%
- borderSize (arrow) : 0
- lastSectionWithArrow (arrow) : False)
- FirstSectionWithArrow (arrow) : False
- borderSize (section) : 0

Pour plus d'information sur ces celles-ci, voir le chapitre sur la liste des propriétés.

De plus, toutes les sections sont passés en type « standard » excepté celle de type « current » qui correspond à la section dont l'identifiant est stocké dans l'instance (id de la dernière section sélectionnée).

Toutes les valeurs que vous auriez pu définir pour ces propriétés seront surchargés par celles prédéfinies.

## 1.5 Cycle de vie d'un Breadcrumb

### 1.5.1 Phase 1 : initialisation et paramétrage

La première phase consiste à définir les propriétés du Breadcrumb qui va être affiché. Pour cela, le composant offre une méthode « New AJUI_BreadCrumb » générant un objet de base représentant une définition par défaut. Libre ensuite à l'utilisateur d'utiliser les fonctions membres attachées à l'objet afin de personnaliser ses propriétés à sa guise et d'ajouter le nombre de section qu'il souhaite.

### 1.5.2 Phase 2 : Calculs et génération du breadcrumb

La deuxième phase qui est la plus conséquente pour le composant est la création du Breadcrumb basée sur sa définition. Durant cette phase, il va se charger de récupérer les propriétés de l'objet afin d'effectuer une série de calcul et de contrôle afin de générer/affecter les différents éléments qui vont composer le Breadcrumb. C'est également dans cette phase que les exceptions liées à l'état et le type de section sont traités. Si un callback associé à un événement (On Clicked, On Double clicked) est assigné à une section et que l'événement est détecté, alors il sera exécuté à la fin de la génération.

### 1.5.3 Phase 3 : Affichage et masquage

À la fin de la génération, l'image est associée à l'objet de formulaire image qui lui a été associée. La gestion de l'affichage du Breadcrumb peut se faire à l'aide des fonctions membres « Show » et « Hide » ou à l'aide des méthodes 4D permettant de gérer les objets de formulaires.

# 2 Méthodes du composant

`New AJUI_BreadCrumb ( { template_param } ) -> breadcrump`

Cette méthode retourne une variable objet (breadcrump) qui représente une instance d'AJUI_BreadCrumb. Elle contient l'ensemble des propriétés et leurs valeurs par défaut ainsi que les formules (fonctions membres) pour les manipuler.

Il est possible de lui passer en paramètre un objet (template_param) afin d'importer un template existant (fichier JSON). Propriétés attendues de l'objet (template_param) passé en paramètre :

- **templateName** : Correspond au nom du fichier JSON à importer (template). Si le fichier n'est pas trouvé, la méthode retournera une nouvelle instance d'AJUI_Breadcrumb.
- **templatePath** (optionnel) : Vous pouvez préciser un chemin pour récupérer le fichier sinon, le composant cherchera dans le dossier par défaut se trouvant dans les ressources de la base hôte (…/Ressources/AJUI_BreadCrumb_Templates/).

```
C_OBJECT($template_obj;$bc)
$templateName:="MybreadcrumbTemplate.json"
$templatePath:=Get 4D folder(Current resources folder)
$templatePath:=$templatePath+"my_Templates_Folder"+Folder separator
$template_obj:=New object()
$template_obj.templateName:=$templateName
$template_obj.templatePath:=$templatePath
$bc:=New AJUI_BreadCrumb ($template_obj)
```

`AJUI_BC_info -> version`

Cette méthode retourne une chaîne de caractère représentant le numéro de version du composant.

`AJUI_BC_loadTemplates ( { folder_path} ) -> templates_col`

Méthode utilitaire permettant de récupérer dans une collection (templates_col), l'ensemble des fichiers JSON situé dans un dossier. Vous pouvez passer en paramètre le chemin du dossier contenant les templates. Si ce n'est pas le cas, la méthode cherchera dans chemin par défaut (…/Ressources/AJUI_BreadCrumb_Templates/).

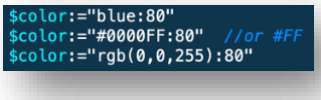
# 3 Listes des propriétés

Un BreadCrumb possède un ensemble de propriétés qui vont permettre de définir sa représentation et ses interactions avec le formulaire hôte.

Dans ce chapitre, nous allons passer en revue les différentes propriétés existantes accessibles par une formule faisant office de **Setter** ou de **Getter**[1].

Toutes les formules pourront être appelées au premier niveau de l'objet.

Concernant les propriétés touchant aux couleurs, vous pouvez leur passer un code couleur sous forme d'une chaîne CSS ( Exemple: "blue" ou "#0000FF" ou "rgb(0,0,255)" )




Par défaut le facteur d'opacité appliqué aux couleurs est de 100%. Pour changer cette valeur à 80 % il suffit d'ajouter cette valeur à la chaîne définissant la couleur en la séparant par le caractère deux points". (Exemple : "blue:80)

Il est recommandé de ne pas modifier manuellement les propriétés dont les valeurs ne sont pas définies par des formules, car elles sont gérées en interne par le composant. Les propriétés liées aux sections seront traitées dans le chapitre de gestion des sections.

## 3.1 Global

Propriétés globales se trouvant au deuxième niveau de l'objet : **MyBC.**breadCrumb.global

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| name | texte | Nom du Breadcrumb et donc de l'objet de formulaire image. | Chaîne vide | **Name** |
| model | texte | Modèle à utiliser : simple, arrow, groupedButtons | arrow | **Model** |
| rtl | booléen | Les sections sont générées de droite à gauche (right to left) | False | **RTL** |
| maxSectionToDraw[2] | entier long | Nombre maximum de sections à afficher | -1 | **MaxSectionToDraw** |

### Global – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **Name** (nom) | - Nom de l'objet de formulaire (texte) |
| **Model** (modèle) | - Nom du modèle (texte) |
| **RTL** (activer) | - Disposition des sections (booléen) |

[1] dans le cas ou aucun paramètre n'est passé.
[2] nombre maximal de section à afficher. Sans limite correspond à -1 qui est la valeur par défaut. Il est possible d'afficher moins de sections que celles créées.

| | |
|---|---|
| **MaxSectionToDraw** (max) | - Nombre maximum de sections à afficher (entier long) |

## 3.2 Box

Propriétés liées au conteneur se trouvant au deuxième niveau de l'objet : **MyBC.**breadCrumb.box

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| bgColor | texte | Couleur de l'arrière-plan. Il est possible de définir un taux d'opacité (%) (Ex: black:80 - noir avec une opacité de 80%) | #F0EEF1 | **BgColor** |
| centered | booléen | Permet de centrer le Breadcrumb par rapport à la position et la taille de l'objet image définit dans l'éditeur de formulaire. Ne s'applique que si dynamicSize est activé. | False | **IsCentered** |
| dynamicSize | booléen | Dimensionne la largeur du Breadcrumb automatiquement en fonction de la taille des sections. | True | **IsDynamicSize** |
| height | entier long | Hauteur du Breadcrumb. Si valeur -1 alors il prend comme hauteur celle définie dans l'éditeur de formulaire. | -1 | **BoxHeight** |
| width | entier long | Largeur du Breadcrumb. Si valeur -1 alors il prend comme largeur celle définie dans l'éditeur de formulaire. | -1 | **BoxWidth** |

### Box – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **BgColor** (couleur) | - Couleur de fond (texte) |
| **IsCentered** (activer) | - Centrage automatique (booléen) |
| **IsDynamicSize** (activer) | - Dimensionnement automatique (booléen) |
| **BoxHeight** (hauteur) | - Hauteur du conteneur (entier long) |
| **BoxWidth** (largeur) | - Largeur du conteneur (entier long) |

## 3.3 Box corner radius

Propriétés liées aux angles du conteneur se trouvant au troisième niveau de l'objet : **MyBC.**breadCrumb.box.cornerRadius

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| left | entier long | Valeur du rayon appliqué aux angles du côté gauche. | 5 | **CornerRadiusLeft** |
| right | entier long | Valeur du rayon appliqué aux angles du côté droit. | 5 | **CornerRadiusRight** |

### Box corner radius – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **CornerRadiusLeft** (rayon) | - Valeur du rayon côté gauche du conteneur (entier long) |
| **CornerRadiusRight** (rayon) | - Valeur du rayon côté droit du conteneur (entier long) |

## 3.4 Box border

Propriétés liées à la bordure du conteneur se trouvant au troisième niveau de l'objet : **MyBC.**breadCrumb.box.border

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| color | texte | Couleur de la bordure du container. Il est possible de définir un taux d'opacité (%) (Ex: black:80 - noir avec une opacité de 80%) | black | **BorderColor** |
| size | entier long | Taille de la bordure du container. | 2 | **BorderSize** |

### Box border – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **BorderColor** (couleur) | - Couleur de bordure (texte) |
| **BorderSize** (taille) | - Taille de bordure (entier long) |

## 3.5 Box padding

Propriétés liées aux paddings du conteneur se trouvant au troisième niveau de l'objet : **MyBC.**breadCrumb.box.padding

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| horizontal | entier long | Padding horizontal. | 0 | **PaddingH** |
| vertical | entier long | Padding vertical. | 0 | **PaddingV** |

### Box padding – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **PaddingH** (padding) | - Padding horizontal (entier long) |
| **PaddingV** (padding) | - Padding verticale (entier long) |

## 3.6 Divider

Propriétés du séparateur se trouvant au deuxième niveau de l'objet : **MyBC.**breadCrumb.divider

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| margin | entier long | Marge du divider. | 0 | **DividerMargin** |
| texte | texte | Caractère(s) à utiliser comme séparateur pour le modèle simple. | > | **Divider** |

### Divider – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **DividerMargin** (marge) | - Marge du divider (entier long) |
| **Divider** (texte) | - Séparateur (texte) |

## 3.7 Divider font

Propriétés du séparateur pour le modèle simple se trouvant au troisième niveau de l'objet : **MyBC.**breadCrumb.divider.font

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| color | texte | Couleur du séparateur. Il est possible de définir un taux d'opacité (%) (Ex: black:80 - noir avec une opacité de 80%) | blue | **DividerFontColor** |
| name | texte | Nom de la police à utiliser. Plusieurs polices peuvent être définies en les séparant par une virgule sans espace. Le composant recherchera la première police utilisable par rapport à votre OS.<br>Attention, certaines polices abaissent certain caractères et donc ceux-ci ne pourront pas être verticalement centrés au niveau du Breadcrumb. | Arial | **DividerFontName** |
| size | entier long | Taille du séparateur. | 24 | **DividerFontSize** |
| style | texte | Style à appliquer au séparateur. Disponible : bold, italic, underline, strikethrough. Il est possible d'utiliser plusieurs styles en les séparant par une virgule sans espace. | none | **DividerFontStyle** |

### Divider font – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **DividerFontColor** (couleur) | - Couleur de la police (texte) |
| **DividerFontName** (police) | - Nom de la police (texte) |
| **DividerFontSize** (taille) | - Taille de la police (entier long) |
| **DividerFontStyle** (style) | - Style de la police (texte) |

## 3.8 Divider arrow

Propriétés pour le modèle arrow se trouvant au troisième niveau de l'objet : **MyBC.**breadCrumb.divider.arrow

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| borderSize | entier long | Épaisseur de la flèche. | 4 | **ArrowBorderSize** |
| color | texte | Couleur de la flèche. Il est possible de définir un taux d'opacité (%) (Ex: black:80 - noir avec une opacité de 80%) | #0048AA | **ArrowColor** |
| firstSectionWithArrow | booléen | Si faux, la largeur de la flèche est à zéro pour la première section. | True | **IsFirstSectionWithArrow** |
| lastSectionWithArrow | booléen | Si faux, la largeur de la flèche est à zéro pour la dernière section. | False | **IsLastSectionWithArrow** |
| paddingH | entier long | Padding appliquer au niveau du contenu de la section dans un modèle arrow | 16 | **ArrowPaddingH** |
| width | texte | Largeur de la flèche. | 50% | **ArrowWidth** |

### Divider arrow – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **ArrowBorderSize** (épaisseur) | - Épaisseur de la flèche (entier long) |
| **ArrowColor** (couleur) | - Couleur de la flèche (texte) |
| **IsFirstSectionWithArrow** (activer) | - Flèche à la première section (booléen) |
| **IsLastSectionWithArrow** (activer) | - Flèche à a la dernière section (booléen) |
| **ArrowPaddingH** (padding) | - Padding de la section (entier long) |
| **ArrowWidth** (largeur) | - Largeur de la flèche (entier long) |

## 3.9 Section types

Propriétés liées aux états et types de section se trouvant au cinquième niveau de l'objet : **MyBC.**breadCrumb.sectionTypes[Type][State]

| Nom | Type | Description | Par défaut | Formule |
|---|---|---|---|---|
| bgColor | texte | Couleur de l'arrière-plan. Il est possible de définir un taux d'opacité (%) (Ex: black:80 - noir avec une opacité de 80%) | #D4E3FE | **SectionBgColor** |
| borderColor | texte | Couleur de la bordure. | Black | **SectionBorderColor** |
| borderSize | entier long | Taille de la bordure. | 0 | **SectionBorderSize** |
| fontColor | texte | Couleur de la police. Il est possible de définir un taux d'opacité (%) (Ex: black:80 - noir avec une opacité de 80%) | #0148AA | **SectionFontColor** |
| fontName | texte | Nom de la police. | Arial | **SectionFontName** |
| fontSize | entier long | Taille de la police. | 12 | **SectionFontSize** |
| fontStyle | texte | Style de la police. Disponible : bold, italic, underline, strikethrough. Il est possible d'utiliser plusieurs styles en les séparant par une virgule sans espace. | None | **SectionFontStyle** |
| pictureOpacity | entier long | Pourcentage d'opacité des images. | 100 | **PictureOpacity** |
| replacingColor | texte | Couleur devant remplacer la couleur actuelle dans une image de type SVG. | Chaîne vide | **ReplacingColor** |

## Section types – Formules et leurs paramètres

| Nom de formule | Paramètre(s) |
|---|---|
| **SectionBgColor** (constante ; couleur) | - Type et état (constante)<br>- Couleur de fond (texte) |
| **SectionBorderColor** (constante ; couleur) | - Type et état (constante)<br>- Couleur de bordure (texte) |
| **SectionBorderSize** (constante ; taille) | - Type et état (constante)<br>- Taille de bordure (entier long) |
| **SectionFontColor** (constante ; couleur) | - Type et état (constante)<br>- Couleur de la police (texte) |
| **SectionFontName** (constante ; police) | - Type et état (constante)<br>- Nom de la police (texte) |
| **SectionFontSize** (constante ; taille) | - Type et état (constante)<br>- Taille de la police (entier long) |
| **SectionFontStyle** (constante ; style) | - Type et état (constante)<br>- Style de la police (texte) |
| **PictureOpacity** (constante ; pourcentage) | - Type et état (constante)<br>- Pourcentage d'opacité (entier long) |
| **ReplacingColor** (constante ; couleur) | - Type et état (constante)<br>- Couleur de remplacement (texte) |

# 4 Gestion des sections

Dans AJUI_BreadCrumb, une **section** est un objet contenant un ensemble de propriétés affectant uniquement celle-ci. L'ensemble de ces objets sections sont regroupés dans une collection au niveau de l'instance.

## 4.1 Description et utilisation d'une section

Une section est créée et manipulée à l'aide d'un identifiant (id) permettant au composant de la reconnaître au moment où la souris interagira avec celle-ci. Un attribut position lui est associé afin de définir l'ordre dans lequel elle sera positionnée et affichée dans le breadcrumb.

Chaque section est également associée à un type (standard, first, previous, next, current). Le type va définir quelles propriétés d'états (Ex : AJUI_bc_standard_hover) que le composant doit appliquer pour cette section.

Chaque section possède un contenu textuel (label) et/ou une image spécifique, un callback et un objet contenant les paramètres à passer au callback. Un callback peut être assigné à une section sur les événements « On Clicked » et/ou « On DoubleClicked »

Pour finir, une section possède une propriété (disable) permettant de la faire passer d'active à inactive.

Dans les deux sous-chapitres suivants, nous allons présenter les fonctions membres disponibles pour gérer les sections et également les propriétés qui sont associées à chaque section.

## 4.2 Fonctions membres

La gestion des sections se fait via des fonctions membres CRUD ainsi que quelques méthodes utilitaires pour simplifier leur gestion. Voici cette liste :

```
AddSection ( id { ; param } )
```

Permets d'ajouter une section. Elle attend en premier paramètre l'id de la section cible et vous pouvez optionnellement passer en deuxième paramètre un objet contenant une partie ou l'ensemble des propriétés d'une section avec les valeurs que vous souhaitez. Les propriétés non spécifiées recevront leurs valeurs par défaut.

```
UpdateSection ( id { ; param } ) -> { param }
```

Permets de mettre à jour une section. Elle attend en premier paramètre l'ID de la section cible et en deuxième paramètre optionnel un objet contenant les propriétés à modifier avec leurs nouvelles valeurs (Setter). Si vous ne passez pas le deuxième paramètre, la fonction vous retourne la définition de la section dans un objet (Getter).

```
RemoveSection ( id )
```

Permets de supprimer une section. Elle attend l'id de la section cible en paramètre.

```
SetCurrentSection ( id )
```

Fonction utilitaire permettant d'assigner le type "current" à une section. Elle va également appliquer aux sections précédentes le type "previous" et aux sections suivantes, le type "next". Ce traitement ne s'appliquera pas aux sections de type "first". Cette fonction attend l'id de la section cible en paramètre.

```
GetNextPosition -> nextPosition
```

Retourne dans un entier long la prochaine valeur de position (valeur la plus grande plus un).

`RemoveAllSections`

Retire toutes les sections de la collection.

## 4.3 Liste des propriétés d'une section

Comme vous avez pu l'apercevoir dans les fonctions membres, l'ensemble des propriétés ci-dessous peuvent être modifiées en les passant dans un objet dans les fonctions membres CRUD. Si ce n'est pas le cas, les valeurs par défaut seront appliquées (sauf pour l'id qui est obligatoire lors de la création d'une section)

| Nom | Type | Description | Par défaut |
|---|---|---|---|
| id | texte | id de la section. Elle n'a pas de valeur par défaut, car elle doit **obligatoirement** être renseignée lors de l'utilisation des fonctions CRUD. | - |
| position | texte | Position de la section dans le Breadcrumb. Par défaut, le composant recherche la valeur la plus haute des positions et ajoutera un. | Valeur de position la plus grande +1 |
| type | texte | La section est reliée à un type parmi : standard, first, current, previous et next. | standard |
| label | texte | Contenu textuel de la section. | section + valeur position |
| disable | booléen | Si True (vrai), la section est considérée dans l'état disable. | False |
| picturePath | texte | Chemin de dossier à partir du dossier « ressources » de la base hôte. Vous pouvez également utiliser « # » plus le nom de votre fichier image comme chemin ( valable uniquement avec un template AJUI BreadCrumb importé si la ou les images associées sont dans le même dossier que le template) | Chaîne vide |
| picturePosition | texte | Position de l'image par rapport au texte. Valeur possible : left, right. | left |
| pictureMargin | texte | Marge entre le texte et l'image. | 10 |
| pictureHeight | texte | Taille de l'image en hauteur, le nombre doit être suivi de « px » si vous voulez définir une taille en pixels sinon passez une valeur en % si vous voulez un pourcentage par rapport à la hauteur disponible de la section.<br>La largeur est redimensionnée automatiquement en fonction de la hauteur calculée basée sur la taille de la police. | 40% |
| colorToReplace | texte | Couleur actuelle d'un SVG. Ici c'est la valeur de la balise « fill ». Utilisable uniquement avec une image dont l'extension est SVG (voir la propriété replacingColorSVG). | Chaîne vide |
| onClickCB | texte | Nom de la méthode de callback sur l'événement « On Clicked ». | Chaîne vide |
| onClickCBParams | objet | Object contenant les paramètres du callback sur l'événement « On Clicked » | null |
| onDoubleClick | texte | Nom de la méthode de callback sur l'événement « On Double Clicked ». | Chaîne vide |
| onDoubleClickCBParams | objet | Object contenant les paramètres du callback sur l'événement « On Double Clicked » | null |

# 5 Fonctions membres non liées à une propriété

Liste des fonctions membres n'étant pas attachées à une propriété.

`Draw ()`

Génère le BreadCrumb en prenant en compte l'état actuel pour chaque section.

`Export (templateName { ; templateFolderPath } )`

Permets de créer un template de votre objet AJUI_Breadcrumb dans un fichier au format JSON. La formule requiert en premier paramètre le nom à associer au template (nom du fichier). Un deuxième paramètre optionnel vous permet de spécifier le chemin de dossier, sinon il sera exporté dans le dossier par défaut (…/Ressources/AJUI_BreadCrumb_Templates/).

`Hide ()`

Cache le Breadcrumb.

`ResetAllStates`

Permets de supprimer l'ensemble des exceptions.

`ResetState ( state )`

Permets de supprimer l'ensemble des exceptions d'un état. Vous devez passer la constante correspondant à l'état en paramètre.

`RemovePropertyException ( property ; state)`

Permets de supprimer une exception liée à une propriété. Le nom de la propriété est passé en premier paramètre et la constante correspondant à l'état en deuxième paramètre. (Voir chapitre 1.3.3 concernant les gestion des exceptions)

`RemovePropertyExceptions ( property )`

Permets de supprimer l'ensemble des exceptions liées à une propriété pour chacun des states. Le nom de la propriété est attendu en paramètre. (Voir chapitre 1.3.3 concernant les gestion des exceptions)

`RemovePropertyTypeExceptions ( property ; typeSection )`

Permets de supprimer l'ensemble des exceptions liées à une propriété, mais uniquement pour un type défini. Le nom de la propriété est attendu en premier paramètre et le type en deuxième paramètre (disponible : standard | first | current | next | previous). (Voir chapitre 1.3.3 concernant les gestion des exceptions)

`Show ()`

Affiche le Breadcrumb.

`RemoveCurrent ()`

Lors de l'utilisation d'un modèle **groupedButtons**, l'instance conserve l'id de la section selectionnée (type « current »). Cette fonction permet d'effacer l'id en mémoire.

 À noter que les fonctions membres **`Hide`** et **`Show`** sont en fait des wrappers de la méthode 4D « OBJECT SET VISIBLE ». Vous pouvez très bien l'utiliser directement si vous le souhaitez.

 Concernant les fonctions membres permettant de retirer une ou plusieurs exceptions, il faut bien assimiler qu'elles n'auront pas de répercussions pour les propriétés liées à « standard-default », car ces propriétés sont les valeurs par défaut et ne sont donc pas des exceptions.

# 6 Prise en main

## 6.1 Installation

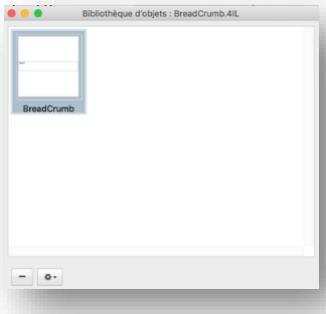
AJUI_BreadCrumb doit être placé dans le dossier composant de votre application.

## 6.2 Principes d'utilisation

Dans cette partie, nous décrivons la suite d'opération à réaliser afin de générer un simple Breadcrumb dans le contexte du formulaire principal.

### 6.2.1 Prérequis

AJUI_Breadcrumb a besoin d'un objet de type image qui va servir de modèle. Cet objet est fourni avec la librairie "BreadCrumb.4IL" que vous pouvez ouvrir depuis votre application (/File/Open/Object Library...) et il vous suffira de faire glisser (Drag & Drop) l'objet « BreadCrumb_object » sur votre formulaire.




Vous pourrez dupliquer cet objet pour créer autant de Breadcrumb que nécessaire. Chaque Breadcrumb devra avoir un nom spécifique que vous pouvez définir à votre fantaisie.

Si vous créez le Breadcrumb par vous-même, n'oubliez pas d'activer les événements requis au fonctionnement du Breadcrumb (voir le chapitre sur les événements).

### 6.2.2 Base de création d'un breadcrumb

La première chose à faire lors de la création d'un Breadcrumb est d'ouvrir la méthode de votre objet de formulaire image et de créer une instance avec la méthode « New AJUI_BreadCrumb ». Ensuite, associer le nom de votre objet de formulaire à l'instance avec la fonction membre « Name ». Puis, ajouter des sections avec la fonction membre « AddSection ». Voilà, le Breadcrumb est créé et va utiliser les valeurs par défaut, il ne vous reste plus qu'à adapter les paramètres de celui-ci à vos besoins en termes d'interface et de fonctionnement.

```
Case of
: (Form event=On Load)

  Form.bc:=New AJUI_BreadCrumb ()
  Form.bc.Name("myBC")
  Form.bc.AddSection("sect_home";New object("label";"Home"))
  Form.bc.AddSection("sect_products";New object("label";"Products"))
  Form.bc.AddSection("sect_enterprise";New object("label";"Enterprise"))
  Form.bc.AddSection("sect_support";New object("label";"Support"))
  Form.bc.AddSection("Store";New object("label";"Store"))
  Form.bc.AddSection("sect_contact";New object("label";"Contact"))

End case

Form.bc.Draw()
```


Pour cela, il suffit d'appeler les fonctions membres des propriétés qui sont décrites dans les chapitres correspondants. Tester, les différentes compositions et types. Changer également les valeurs des propriétés pour différents états en utilisant les constantes voulues. Pour finir, encapsuler le tout dans un événement « on load » et lancer la fonction « Draw ».

Pour information, les valeurs par défaut font en sorte de réaliser un modèle « arrow ».

Nous avons fait le tour au niveau des bases de construction et pour la suite, nous vous conseillons grandement d'utiliser l'application « AJUI_BreadcrumbLab » qui vous est fournie sous forme de code source et qui sert de démonstrateur du composant.

AJUI_BreadcrumbLab vous met à disposition un environnement vous permettant de mieux appréhender les résultats graphiques des différentes propriétés et de créer rapidement des templates afin de vous simplifier la définition de vos Breadcrumb. Cela permet aussi de voir de façon clair quelles propriétés sont utilisables par chaque modèle. Il est livré avec un certain nombre de modèles prédéfinis.

# 7 Conclusion

Le but de ce document était de vous présenter les principes théoriques du composant ainsi que les différentes méthodes, formules et propriétés à votre disposition pour pouvoir générer des breadcrumbs.

Quant aux éléments pratiques présentés, ils ont pour but de vous permettre de mettre un premier pied à l'étrier et d'aborder quelques cas particuliers qui pourraient intervenir dans l'utilisation du composant.

Si vous désirez une aide pour l'implémentation du composant AJUI_BreadCrumb dans votre application. Vous désirez modifier ou étendre ses fonctionnalités pour un usage spécifique. Vous désirez disposer du code source du composant AJUI_BreadCrumb afin de pérenniser son usage dans votre application avec les futures versions de 4D. N'hésitez pas à nous contacter pour en discuter.