

# La section [Code] de InnoSetup

par [ThierryAIM \(home\)](#)

Date de publication : 27/01/2008

Dernière mise à jour :

**InnoSetup** est un logiciel gratuit de déploiement bien connu des développeurs, pour sa simplicité d'utilisation.

Toutefois, il offre des possibilités extraordinaires, à qui veut bien se pencher sur ses rouages, et particulièrement la section [Code], qui permet de faire *presque* tout ce que l'on veut !

Ce tuto n'est pas exhaustif et n'a pas la prétention de remplacer l'aide d'**InnoSetup**, très bien faite, mais malheureusement, uniquement en anglais

## I - Introduction

Pré requis indispensables :

## II - La section [Code]

Que peut-on faire avec la section [Code] de InnoSetup ?

## III - Les fonctions d'évènements

III-1 - Les évènements standards

III-2 - Les évènements personnalisés

## IV - Utiliser le code dans la partie script

IV-1 - Exemple 1 : Ajouter un lien sur toutes les pages du script

IV-2 - Exemple 2 : Ajouter un bouton 'action' sur toutes les pages du script

## V - Créer des pages personnalisées

V-1 - Les pages pré formatées de InnoSetup

V-1-1 - Exemple 1 : Insérer une page de sélection de répertoires

V-1-2 - Exemple 2 : Insérer un page de sélection d'options

V-1-3 - Les pages personnalisées (custom page)

## VI - Renvoyer les informations au script : fonctions de retour

VI-1 - Exemple 1 : Retour d'informations de type String

VI-2 - Exemple 2 : Retour d'informations de type Boolean

## VII - Utiliser les informations du script dans le code


Exemple d'utilisation pour un installateur multilingue :

## VIII - Pour finir ...

VIII-1 - Liens

VIII-2 - Remerciements

## I - Introduction

Tout d'abord, si vous n'avez pas ou ne connaissez pas **InnoSetup**, pour pouvez le télécharger gratuitement sur  **le site de l'éditeur** ou par l'intermédiaire de la pages **Outils** du **forum VB6**

Le principal handicap de **InnoSetup**, est, pour la communauté francophone, d'être publié et documenté exclusivement en anglais.

Certaines explications, publiés ci-après, sont extraites soit :

- des codes exemples, fournis avec l'installation de InnoSetup
- de la documentation du logiciel, traduite en français par mes soins

Il n'est pas question de détailler ici toutes les fonctions de la **VCL** de **InnoSetup**, mais d'essayer de comprendre comment utiliser la section **[Code]**.

Les fonctions et procédures de la VCL seront à rechercher dans l'aide, suivant ce que vous souhaitez exécuter.

J'adresse ici un tout grand **Merci** aux concepteurs de ce logiciel, dont je suis l'un des plus grand fan.

J'espère que ce petit tutoriel sans prétention vous aidera à utiliser au mieux la puissance de ce logiciel.

Pour toutes informations complémentaires, je vous engage à consulter l'aide d'**InnoSetup**, très bien faite, pour qui maîtrise la langue de Shakespeare.

Tout ce que je sais de ce logiciel, je l'ai appris de là, des exemples fournis lors de l'installation ... *(et de quelques prises de tête !)*

## Pré requis indispensables :

- une bonne pratique du logiciel **InnoSetup** (connaissance des autres sections standard de script)
- un minimum de connaissance du langage Pascal ou Delphi (principe et syntaxe)

## II - La section [Code]

 La section **[Code]** d'un script **InnoSetup** utilise le langage **Delphi**, le logiciel étant lui-même écrit avec ce langage.

Je ne m'attarderai pas sur la syntaxe de ce langage, ni sur la partie script standard de InnoSetup. (cf. Pré requis)

Nous ne traiterons pas non plus dans ce tutoriel, de l'extension **InnoSetup Préprocesseur**, qui fera (plus tard ...) l'objet d'un prochain cours.

## Que peut-on faire avec la section [Code] de InnoSetup ?

A peu près tout ce que l'on veut lors de l'installation d'un progiciel !

- Modifier les pages standard
- Ajouter des pages pré formatées
- Créer de toutes pièces des pages personnalisées
- Dialoguer avec le script
- Créer ses fonctions personnalisées
- ....

Les chapitres suivants vont essayer de vous aider à tirer partie du meilleur de ces fonctionnalités.

## III - Les fonctions d'évènements

La section **[Code]** est dite événementielle, c'est à dire qu'elle réagit aux interruptions d'évènements, comme tout langage de ce type (VB, Delphi, ...)

### Les évènements sont de 2 types :

- Les évènements standards : générés automatiquement par le script
- Les évènements personnalisés : ceux que vous allez créer

### III-1 - Les évènements standards

Certains évènements sont déclenchés lors du déroulement du script.

Utilisez les fonctions d'évènements afin de modifier le comportement de votre installateur.

Je ne vais pas récrire entièrement l'aide de [InnoSetup](#), mais la liste ci-dessous vous permettra d'avoir une idée de quoi et où chercher :

### Les évènements standards d'installation de InnoSetup :

- **function InitializeSetup(): Boolean;** : appelé lors de l'initialisation du setup (c'est le 1er évènement qui se déclenche)
- **procedure InitializeWizard();** : procédure d'initialisation générale
- **procedure DeinitializeSetup();** : appelé en fin d'installation (c'est le dernier évènement qui se déclenche)
- **procedure CurStepChanged(CurStep: TSetupStep);** : utilisez cet évènement afin de réaliser vos propres pré-install et post-install
- **function NextButtonClick(CurPageID: Integer): Boolean;** : se déclenche lors de l'appui sur le bouton Suivant
- **function BackButtonClick(CurPageID: Integer): Boolean;** : se déclenche lors de l'appui sur le bouton Précédent
- **procedure CancelButtonClick(CurPageID: Integer; var Cancel, Confirm: Boolean);** : se déclenche lors de l'appui sur le bouton Annuler
- **function ShouldSkipPage(PageID: Integer): Boolean;** :
- **procedure CurPageChanged(CurPageID: Integer);** : se déclenche lors de l'affichage d'une nouvelle page
- **function CheckPassword(Password: String): Boolean;** : si présente dans le code, provoque l'affichage d'un boîte de saisie d'un mot de passe
- **function NeedRestart(): Boolean;** : retourne Vrai afin d'informer l'utilisateur que la désinstallation nécessite un redémarrage
- **function UpdateReadyMemo(Space, NewLine, MemoUserInfoInfo, MemoDirInfo, MemoTypeInfo, MemoComponentsInfo, MemoGroupInfo, MemoTasksInfo: String): String;**
- **procedure RegisterPreviousData(PreviousDataKey: Integer);** :
- **function CheckSerial(Serial: String): Boolean;** : si présente dans le code, provoque l'affichage d'un champ de saisie d'un numéro de série sur la page UserInfo
- **function GetCustomSetupExitCode: Integer;** :
- 

### Les évènements standards de désinstallation de InnoSetup :

- **function InitializeUninstall(): Boolean;** : retourne Faux pour abandonner la désinstallation, sinon Vrai
- **procedure DeinitializeUninstall();** : se déclenche en fin de désinstallation

- **procedure CurUninstallStepChanged(CurUninstallStep: TUninstallStep);** : identique à CurStepChanged, mais lors de la désinstallation
- **function UninstallNeedRestart(): Boolean;** : retourne Vrai afin d'informer l'utilisateur que la désinstallation nécessite un redémarrage

Le projet standard '**CodeExample1.iss**' montre comment utiliser la plupart de ces évènements

## III-2 - Les évènements personnalisés

Il est bien sûr possible de créer ses propres procédures ou fonctions d'évènement afin de les affecter aux contrôles des pages personnalisées que nous allons construire par la suite.

Exemple de procédure d'évènement pour le click d'un bouton :


```
[Code]
procedure ButtonOnClick(Sender: TObject);
begin
  MsgBox('Vous avez cliquer sur le bouton !', mbInformation, mb_Ok);
end;
```

La liaison du contrôle et de son évènement se fera par l'intermédiaire d'une propriété d'évènement et de l'opérateur @

Lier la procédure d'évènement au contrôle :

```
Button.OnClick := @ButtonOnClick;
```

Vous trouverez ci-après des exemples complets d'utilisation des fonctions d'évènements personnalisés.

 **A l'attention des "Delphistes"** (suggéré par mon compère [sjrd](#)) :

*Les procédures et fonctions définies dans la section [Code] sont en réalité des méthodes (et non des routines) : **procedure of object** ou **function of object**.*

*C'est pour cela que vous pouvez les assigner aux évènements standard des composants.*

*Toutefois, le Self n'est pas accessible : il est utilisé en interne pour contenir des données spéciales, dont vous n'avez pas envie de connaître la nature, croyez-moi... ^*

## IV - Utiliser le code dans la partie script

Dans ce chapitre, nous allons voir comment utiliser la section **[Code]** afin de modifier les pages générées par script de base.

### IV-1 - Exemple 1 : Ajouter un lien sur toutes les pages du script

Cet exemple est adapté du projet standard "CodeClass.iss".

#### **But :**

Ajouter un lien cliquable sur toutes les pages.

#### **Explications :**

- Créer un Label auquel on va donner l'apparence d'un lien à l'aide des propriétés *Caption*, *Cursor*, *Font.Style*, *Font.Color*
- Associer à ce Label une procédure d'événement (*URLLabelOnClick*) à l'aide de la propriété *OnClick*

#### Procédures d'événements

```
[Code]
Procedure URLLabelOnClick(Sender: TObject);
var
  ErrorCode: Integer;
begin
  ShellExec('open', 'http://www.developpez.com', '', '', SW_SHOWNORMAL, ewNoWait, ErrorCode);
end;
```

- La procédure standard *InitializeWizard* sera placée de préférence en fin de code, et de toute façon après toutes les fonctions ou procédures auxquelles elle fait appel :

#### Procédure d'initialisation

```
[Code]
{*** INITIALISATION ***}
Procedure InitializeWizard;
var
  URLLabel: TNewStaticText;
begin
  URLLabel := TNewStaticText.Create(WizardForm);
  URLLabel.Caption := 'www.developpez.com';
  URLLabel.Cursor := crHand;
  URLLabel.OnClick := @URLLabelOnClick;
  URLLabel.Parent := WizardForm;
  { Alter Font *after* setting Parent so the correct defaults are inherited first }
  URLLabel.Font.Style := URLLabel.Font.Style + [fsUnderline];
  URLLabel.Font.Color := clBlue;
  URLLabel.Top := WizardForm.ClientHeight - URLLabel.Height - 15;
  URLLabel.Left := ScaleX(20);
end;
```

### IV-2 - Exemple 2 : Ajouter un bouton 'action' sur toutes les pages du script

#### Un bouton Infos sur toutes les pages ...

```
[Code]
```

### Un bouton Infos sur toutes les pages ...

```
procedure InfosButtonOnClick(Sender: TObject);
begin
  MsgBox('Vous avez cliquer sur le bouton Infos...', mbInformation, mb_Ok);
end;

procedure InitializeWizard();
var
  InfosButton, CancelButton: TButton;
begin

  CancelButton := WizardForm.CancelButton;

  InfosButton := TButton.Create(WizardForm);
  InfosButton.Left := WizardForm.ClientWidth - CancelButton.Left - CancelButton.Width;
  InfosButton.Top := CancelButton.Top;
  InfosButton.Width := CancelButton.Width;
  InfosButton.Height := CancelButton.Height;
  InfosButton.Caption := '&Infos...';
  InfosButton.OnClick := @InfosButtonOnClick;
  InfosButton.Parent := WizardForm;
end;
```

## V - Créer des pages personnalisées


Dans ce chapitre, nous allons voir comment utiliser la section **[Code]** afin de créer des pages supplémentaires à insérer parmi les pages générées par script de base.

### V-1 - Les pages pré formatées de InnoSetup

InnoSetup permet de générer des pages pré formatées, c'est à dire prévues pour une acquisition d'informations bien précises :

#### Liste non exhaustive des type de pages préformatées de InnoSetup :

- **TInputDirWizardPage** : Page contenant des textboxes et des boutons "Parcourir" pour sélectionner des répertoires
- **TInputFileWizardPage** : Page contenant des textboxes et des boutons "Parcourir" pour sélectionner des fichiers
- **TInputOptionWizardPage** : Page contenant des "Checkboxes" ou de "Radio buttons"
- **TInputQueryWizardPage** : Page contenant des textboxes pour saisie d'informations textes
- **TWizardPage** : Page entièrement personnalisée

 Vous ne pouvez pas modifier la forme de ces pages, uniquement ajouter des champs de saisie, correspondant au type d'information attendue sur chaque page.

#### V-1-1 - Exemple 1 : Insérer une page de sélection de répertoires

Cet exemple est extrait d'un projet personnel.

Pour réaliser cette exercice, nous allons utiliser la fonction **CreateInputDirPage**

#### La fonction **CreateInputDirPage** :


```
function CreateInputDirPage(const AfterID: Integer; const ACaption, ADescription, ASubCaption: String; AAppendDir: Boolean; ANewFolderName: String): TInputDirWizardPage;
```

le paramètre **AfterID**, le plus important, spécifie la page **après** laquelle vous voulez insérer la page créée

#### Liste des ID des pages standards de InnoSetup :

- **wpWelcome** : page 'Accueil'
- **wpLicense** : page info Licence
- **wpPassword** : page de saisie d'un mot de passe
- **wpInfoBefore** : page information avant installation
- **wpUserInfo** : page saisie des informations utilisateur
- **wpSelectDir** : page de sélection du répertoire d'installation
- **wpSelectComponents** : page de sélection des composants à installer
- **wpSelectProgramGroup** : page de sélection du groupe de programme du menu démarrer
- **wpSelectTasks** : pages de sélection des tâches supplémentaires
- **wpReady** : page info 'prêt à installer' (ReadyToInstall)

- *wpPreparing* : page info 'préparation en cours'
- *wplInstalling* : page installation en cours
- *wplInfoAfter* : page information après installation
- *wpFinished* : page 'Fin'

 *Toutes ces pages ne seront par obligatoirement présente dans votre setup. Cela dépend des options choisies dans votre script de base.*

Pour insérer une page à la suite d'une autre page déjà créée par le code, spécifiez l'ID de votre page précédente.

#### Exemple d'insertion à la suite d'une page standard

```
[Code]
//La page sera inserée après la page d'info Licence
PageParam := CreateInputDirPage(wpLicense, ...);
```

#### Exemple d'insertion à la suite d'une autre page personnelle

```
[Code]
//La page sera inserée après la page perso PageInfos
PageParam := CreateInputDirPage(PageInfo.ID, ...);
```

Utiliser la méthode **Add** pour ajouter des *textbox* avec un bouton "Parcourir"

**NOTA** : La mise en page est automatique, définie par le schéma de la page pré formatée

Utiliser la propriété **Values** pour définir ou accéder aux valeurs.

#### Procédure de création des pages personnalisées

```
[Code]
// Variables Globales
var
  PageParam: TInputDirWizardPage;

// Creer les Pages Personnalisées
procedure CreateTheWizardPages;
begin
  PageParam := CreateInputDirPage(PageInfo.ID,
  'Répertoires d''installation',
  'Définissez les répertoires d''installation des programmes suivants :',
  'Les données seront enregistrées dans les répertoires définis à cette étape : '#13#10#13#10 +
  'Cliquer sur "Suivant" pour continuer. Pour définir un répertoire différent, cliquez sur
  "Parcourir".',
  False, 'New Folder');

  // Ajouter un élément (avec une valeur vide)
  PageParam.Add('Répertoire d''installation de NumTools');
  PageParam.Add('Répertoire d''installation de notepad++');

  // Initialiser les valeurs par défaut (optional)
  PageParam.Values[0] := 'C:\Numtool2';
  PageParam.Values[1] := ExpandConstant('{pf}')+'\notepad++';

end;
```

**La fonction *UpdateReadyMemo* :**

En supplément de cet exercice, voyons la fonction `UpdateReadyMemo`, afin de mettre à jour le cadre de résumé de la page standard 'ReadyToInstall' :

```
function UpdateReadyMemo(Space, NewLine, MemoUserInfoInfo, MemoDirInfo, MemoTypeInfo, MemoComponentsInfo, MemoGroupInfo, MemoTasksInfo: String): String;
```

Si présente dans la section **[Code]**, cette fonction est appelée automatiquement par la page standard 'ReadyToInstall'

Le texte doit être un chaîne unique

Utiliser le paramètre **Space** pour insérer des espaces standards.

Utiliser le paramètre **NewLine** pour définir une nouvelle ligne.

#### Mettre à jour le Mémo de la page 'Ready'

```
[Code]
// Update ReadyMemo
function UpdateReadyMemo(Space, NewLine, MemoUserInfoInfo, MemoDirInfo, MemoTypeInfo, MemoComponentsInfo, MemoGroupInfo, MemoTasksInfo: String): String;
var
  S, S1: String;
begin
  { Fill the 'Ready Memo' with the normal settings and the custom settings }
  S := '';
  S := S + 'Installer x_Edit32_mod.exe dans :' + NewLine;
  S := S + Space + GetDirPath1(S1) + NewLine + NewLine;
  S := S + 'notepad++ est installé dans :' + NewLine;
  S := S + Space + GetDirPath2(S1) + NewLine;

  S := S + MemoDirInfo + NewLine;
  S := S + MemoTasksInfo + NewLine;

  Result := S;
end;
```

Initialisation du Wizard :

#### Initialisation

```
[Code]
{*** INITIALISATION ***}
Procedure InitializeWizard;
begin
  CreateTheWizardPages;
end;
```

#### V-1-2 - Exemple 2 : Insérer un page de sélection d'options

Cet exemple est extrait du projet personnel.

**La fonction `CreateInputOptionPage` :**

```
function CreateInputOptionPage(const AfterID: Integer; const ACaption, ADescription, ASubCaption: String; Exclusive, ListBox: Boolean): TInputOptionWizardPage;
```

Si le paramètre **Exclusive = True**, les CheckBoxes seront remplacés par des Radio Buttons

Si le paramètre **Listbox = True**, les CheckBoxes ou Radio Buttons seront placés dans une ListBox


### V-1-3 - Les pages personnalisées (custom page)

Pour créer une page personnalisée, nous utiliserons la fonction **CreateCustomPage** de type **TWizardPage**.

#### **La fonction CreateCustomPage :**

```
function CreateCustomPage(const AfterID: Integer; const ACaption, ADescription: String): TWizardPage;
```

Cet exemple est adapté du projet standard "CodeClass.iss".

 **Les éléments que nous voulons rendre accessible au script doivent être déclarés **Public****

#### Déclarations publiques

```
[Code]
// Variables Globales
var
  Edit: TEdit;
  CheckBox1, CheckBox2: TCheckBox;
  PasswordEdit: TPasswordEdit;
  Memo: TMemo;
```

Les contrôles sont créés dynamiquement et positionnés sur la page à l'aide de leurs propriétés : *Top*, *Left*, *Width* et *Height*

#### Construction de la CustomPage :

```
[Code]
// Fonctions et procédures d'évènements
procedure ButtonOnClick(Sender: TObject);
begin
  MsgBox('You clicked the button!', mbInformation, mb_Ok);
end;

// Procédure de construction des pages personnelles
procedure CreateTheWizardPages;
// variables locales
var
  Page: TWizardPage;
  Button: TButton;
  Lbl: TLabel;
begin
  { TButton and others }

  Page := CreateCustomPage(wpWelcome, 'Custom wizard page controls', 'TButton and others');

  Button := TButton.Create(Page);
  Button.Width := ScaleX(75);
  Button.Height := ScaleY(23);
  Button.Caption := 'TButton';
  Button.OnClick := @ButtonOnClick; // on affecte la procédure d'évènement au bouton
  Button.Parent := Page.Surface;

  CheckBox1 := TCheckBox.Create(Page);
  CheckBox1.Top := Button.Top + Button.Height + ScaleY(8);
  CheckBox1.Width := Page.SurfaceWidth;
  CheckBox1.Height := ScaleY(17);
  CheckBox1.Caption := 'TCheckBox 1';
  CheckBox1.Checked := True; // la checkbox est cochée par défaut
```

**Construction de la CustomPage :**

```
CheckBox1.Parent := Page.Surface;

CheckBox2 := TCheckBox.Create(Page);
CheckBox2.Top := CheckBox1.Top + CheckBox1.Height + ScaleY(8);
CheckBox2.Width := Page.SurfaceWidth;
CheckBox2.Height := ScaleY(17);
CheckBox2.Caption := 'TCheckBox 2';
CheckBox2.Checked := False; // la checkbox est décochée par défaut
CheckBox2.Parent := Page.Surface;

Edit := TEdit.Create(Page);
Edit.Top := CheckBox2.Top + CheckBox2.Height + ScaleY(8);
Edit.Width := Page.SurfaceWidth div 2 - ScaleX(8);
Edit.Text := 'TEdit';
Edit.Parent := Page.Surface;

PasswordEdit := TPasswordEdit.Create(Page);
PasswordEdit.Left := Page.SurfaceWidth - Edit.Width;
PasswordEdit.Top := CheckBox2.Top + CheckBox2.Height + ScaleY(8);
PasswordEdit.Width := Edit.Width;
PasswordEdit.Text := 'TPasswordEdit';
PasswordEdit.Parent := Page.Surface;

Memo := TMemo.Create(Page);
Memo.Top := Edit.Top + Edit.Height + ScaleY(8);
Memo.Width := Page.SurfaceWidth;
Memo.Height := ScaleY(89);
Memo.ScrollBars := ssVertical;
Memo.Text := 'TMemo';
Memo.Parent := Page.Surface;

Lbl := TLabel.Create(Page);
Lbl.Top := Memo.Top + Memo.Height + ScaleY(8);
Lbl.Caption := 'TLabel';
Lbl.AutoSize := True;
Lbl.Parent := Page.Surface;
end;
```


On n'oublie pas de générer la/les page(s) :

**Initialisation**

```
[Code]
// Initialisation
procedure InitializeWizard();
begin
    CreateTheWizardPages;
end;
```

## VI - Renvoyer les informations au script : fonctions de retour

Afin de pouvoir accéder, soit par le code, soit par le script, aux valeurs de la page créée il nous faut écrire les fonction de retour.

 *Une fonction de retour doit être définie pour chaque valeur à exploiter.*

### VI-1 - Exemple 1 : Retour d'informations de type String

#### D'une page pré formatée :

##### Fonctions de retour d'infos de type String :

```
[Code]
// Fonctions de retour
function GetDirPath1(Param: String): String;
begin
    Result := PageParam.Values[0] + '\bin';
end;
function GetDirPath2(Param: String): String;
begin
    Result := PageParam.Values[1];
end;
```

#### D'une page personnalisée :

##### Fonctions de retour d'infos de type String :

```
[Code]
// Fonctions de retour
function GetParam1(Param: String): String;
begin
    Result:= Edit.Text;
end;
```

Dans la partie script, utiliser la syntaxe : {**code**:LeNomDeMaFonction}

##### Exemples :

```
[Files]
Source: MonFichier.txt; Destdir: {code:GetDirPath1}; Flags: promptifolder

[INI]
Filename: {app}\istest.ini; Section: InstallSettings; Flags: uninstaldeleteaction
Filename: {app}\istest.ini; Section: InstallSettings; Key: Param1; String: {code:GetParam1}
```

### VI-2 - Exemple 2 : Retour d'informations de type Boolean

##### Fonction de retour d'info de type Boolean

```
[Code]
// Fonctions de retour
function GetParam2: Boolean;
begin
    Result:= CheckBox1.Checked;
end;

function GetParam3: Boolean;
begin
    Result:= CheckBox2.Checked;
```


**Fonction de retour d'info de type Boolean**

```
end;
```

Dans la partie script, utiliser la syntaxe : **Check:** LeNomDeMaFonction

**Exemple :**


```
[INI]
Filename: {app}\MyProgram.ini; Section: InstallSettings; Flags: uninsdeletesection
Filename: {app}\MyProgram.ini; Section: InstallSettings; Key: Param2; String: True; Check:
  GetParam2
Filename: {app}\MyProgram.ini; Section: InstallSettings; Key: Param2; String: False; Check: NOT
  GetParam2
```

 *Vous pouvez combiner les valeurs de retour avec tous les opérateurs logiques !*

**Un exemple :**

```
[INI]
Filename: {app}\MyProgram.ini; Section: InstallSettings; Key: Param1; String: True; Check:
  GetParam2 AND (NOT GetParam3)
```

## VII - Utiliser les informations du script dans le code

 Là, c'est un peu plus simple.

Les informations du script, si elles sont accessibles, peuvent être lu dans le code à l'aide de la fonction **ExpandConstant**

```
function ExpandConstant(const S: String): String;
```

### Exemples :

- **{pf}** : le répertoire 'Program File'
- **{userdata}** : le répertoire 'data' de l'utilisateur connecté
- **{cm:...}** : les messages personnalisés (custom message)
- ....

 Reportez-vous à l'aide d'**InnoSetup** pour connaître les constantes prédéfinies

### Exemple d'utilisation pour un installateur multilingue :

Dans le script, définir :

#### la section [Languages] :

```
[Languages]
Name: en; MessagesFile: compiler:Default.isl;
Name: fr; MessagesFile: compiler:Languages\French.isl;
```

La liste des langues pré-formatées est contenue dans le dossier d'installation de Inno-Setup; normalement votre répertoire "C:\Program Files\Inno Setup 5\Languages"

La langue par défaut est *English*, associée au fichier default.isl

#### la section [CustomMessages] :

```
[CustomMessages]
; Français
fr.whoareyou=Qui êtes-vous ?
fr.titleinfos=Informations personnelles
fr.whoareyou=Qui êtes-vous ?
fr.name=Nom
fr.company=Société
fr.whodescription=Entrez votre nom et celui de votre société, et cliquez sur Suivant.

;English
en.whoareyou=Who are you ?
en.titleinfos=Personal Information
en.whoareyou=Who are you ?
en.name=Name :
en.company=Company :
en.whodescription=Please specify your name and the company for whom you work, then click Next.
```

Dans le code, les '**custom messages**' (**cm**) sont interprétés comme des constantes.

Nous utiliserons la fonction **ExpandConstant**, citée précédemment, afin de les exploiter :

Dans la section [Code] :

```
[Code]
//Variables Globales
var
  PageParam: TInputQueryWizardPage;

// Créer les Pages Personnalisées
procedure CreateTheWizardPages;

begin
  // Create the page
  PageParam := CreateInputQueryPage(wpWelcome,
    ExpandConstant('{cm:titleinfos}'), ExpandConstant('{cm:whoareyou}'),
    ExpandConstant('{cm:whodescription}'));

  // Add items (False means it's not a password edit)
  PageParam.Add(ExpandConstant('{cm:name}'), False);
  PageParam.Add(ExpandConstant('{cm:company}'), False);
  PageParam.Values[0] := 'Valeur_1';
  PageParam.Values[1] := 'Valeur_2';
end;

// Fonctions de retour
function GetParam1(Param: String): String;
begin
  Result := PageParam.Values[0];
end;
function GetParam2(Param: String): String;
begin
  Result := PageParam.Values[1];
end;

{*** INITIALISATION ***}
procedure InitializeWizard;
begin
  CreateTheWizardPages;
end;
```

Pour utiliser les constantes prédéfinies dans **InnoSetup**, utiliser leur nom, comme décrit ci-dessus :


```
[Code]
TempVar:=ExpandConstant('{userappdata}') + ExpandConstant('{username}');
```


## VIII - Pour finir ...

Ceci n'est qu'un aperçu d'une infime partie de tout ce que l'on peut faire avec la section **[Code]** d'**InnoSetup**, mais qui vous permettra de débiter dans la personnalisation de vos programmes d'installation.

Je vous invite à examiner les exemples fournis lors de l'installation, généralement situés dans le répertoire ".\Program Files\Inno Setup 5\Examples"

### VIII-1 - Liens

Télécharger la dernière version de  **Inno-Setup**

Télécharger la dernière version de  **IsTools** (le site est en anglais, mais le logiciel propose un package langue en Français)

Dans la même série, voir aussi :  **Distribuer vos applications VB6 avec InnoSetup**

### VIII-2 - Remerciements

Un tout grand **MERCI** à tous ceux qui ont pris quelques minutes de leur temps précieux pour me relire et me conseiller, et tout particulièrement à **bbil** et **sjrd**.

