Utiliser l'OTA (Open Tools API) de Borland Developer Studio 2006

par William WITTWER (www.codegear.com)

Date de publication : 8 mars 2007

Dernière mise à jour :

Dans ce tutorial, je vais vous montrer comment démarrer simplement un projet vous permettant de créer vos propres extensions à l'environnement de développement BDS.
L'architecture permettant de le faire s'appelle OTA (Open Tools API) et est accessible en .NET par conséquent, vous pouvez écrire vos extensions aussi bien en Delphi pour .NET qu'en C#. Aujourd'hui, nous n'utiliserons que le langage Delphi pour .NET.
Ce document regroupe un ensemble d'informations que j'ai collecté à travers de nombreux sites web et j'essaye d'être le plus didactique possible en expliquant chacune des étapes à effectuer pour ne pas rencontrer de problème.

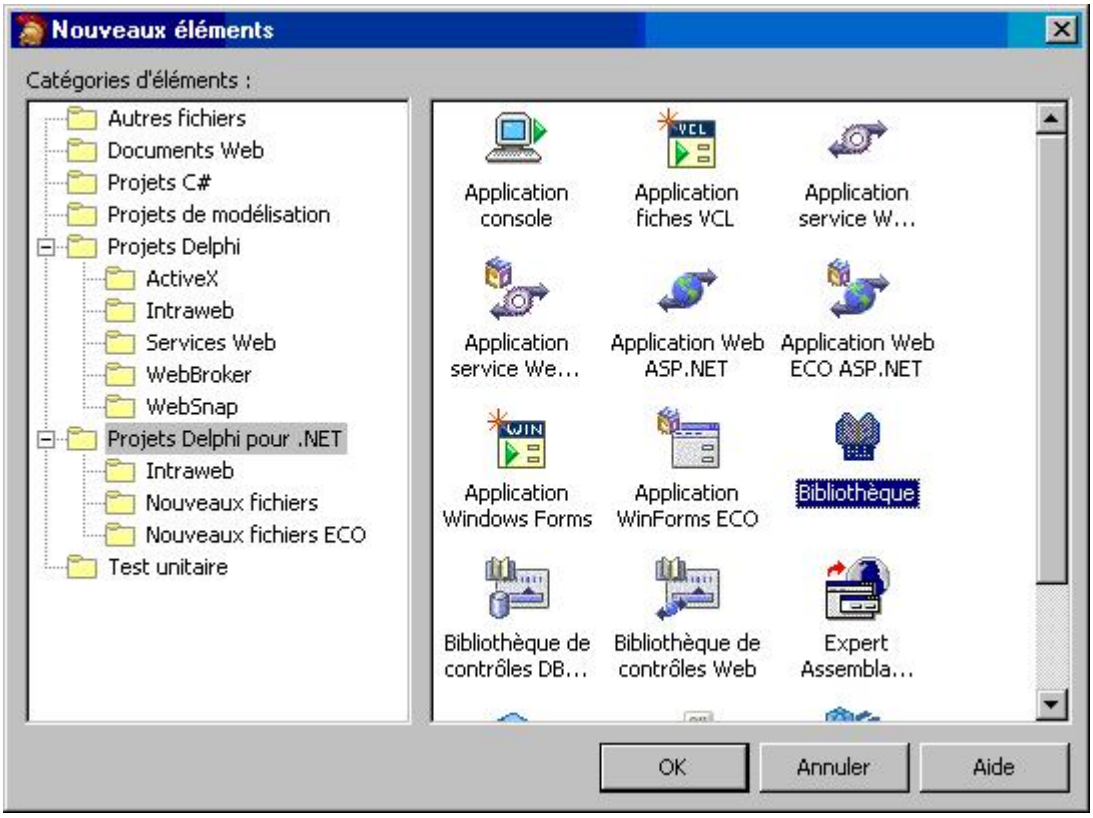
I - Affichage d'une boite de dialogue de message au démarrage de l'EDI
II - Créer un nouvel élément de menu dans l'EDI
III - Affichage de l'extension dans l'écran de démarrage de l'EDI (Splash screen)
IV - Affichage de l'extension dans la fenêtre A propos de...
V - Création d'un élément de menu totalement personnalisé
VI - Utilisation de la fenêtre de messages de BDS
VII - Conclusion

I - Affichage d'une boite de dialogue de message au démarrage de l'EDI

Lancer BDS 2006, Delphi pour Microsoft .NET Framework.

Créer une nouvelle bibliothèque, en cliquant sur :

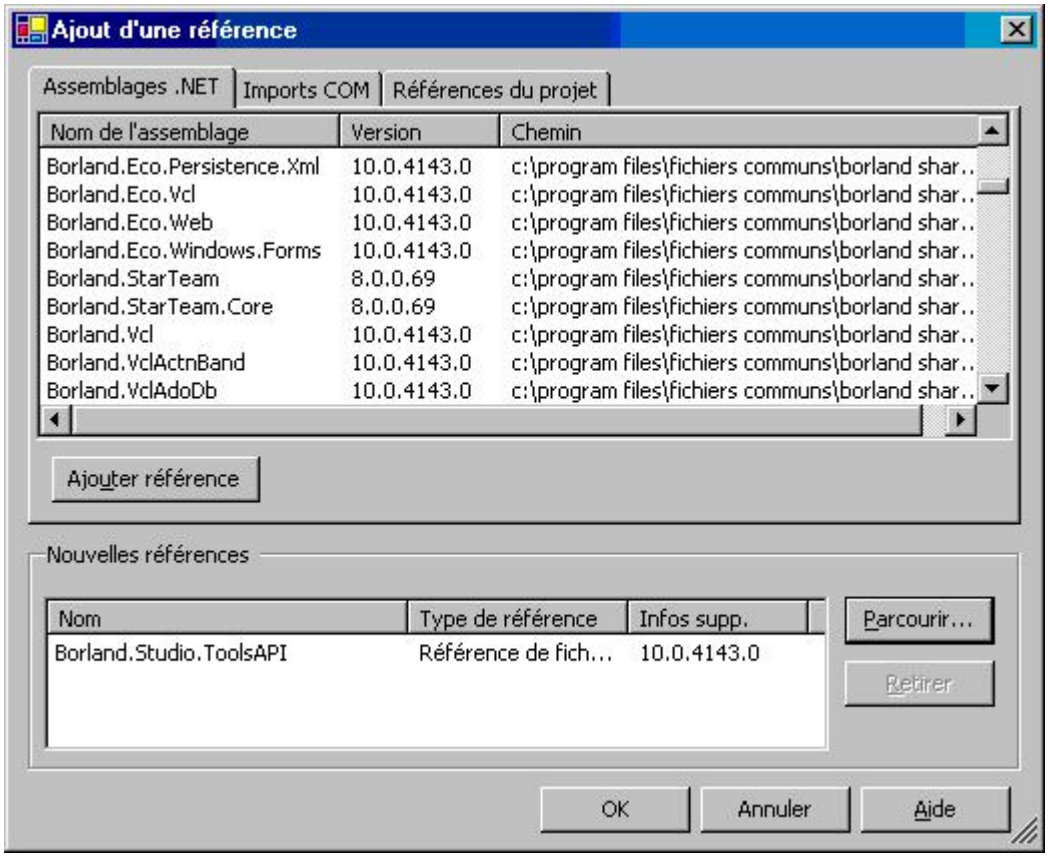
- Menu Fichier/Nouveau/Autre...
- Projets Delphi pour .NET
- Bibliothèque



Sauvegarder le projet (pas de parenthèse dans le chemin)

Ajouter l'assemblage (assembly) pour accéder à l'OTA :

- Menu Projet/Ajouter une référence...
- Choisir l'assemblage : Borland.Studio.ToolsAPI
  - Si vous ne la trouvez pas, cliquez sur le bouton Parcourir...
  - Choisissez le fichier Borland.Studio.ToolsAPI.dll du répertoire bin de votre installation de BDS.
- Cliquez sur le bouton OK.



Modifions le code pour faire que l'extension que nous avons créée fasse quelque chose :

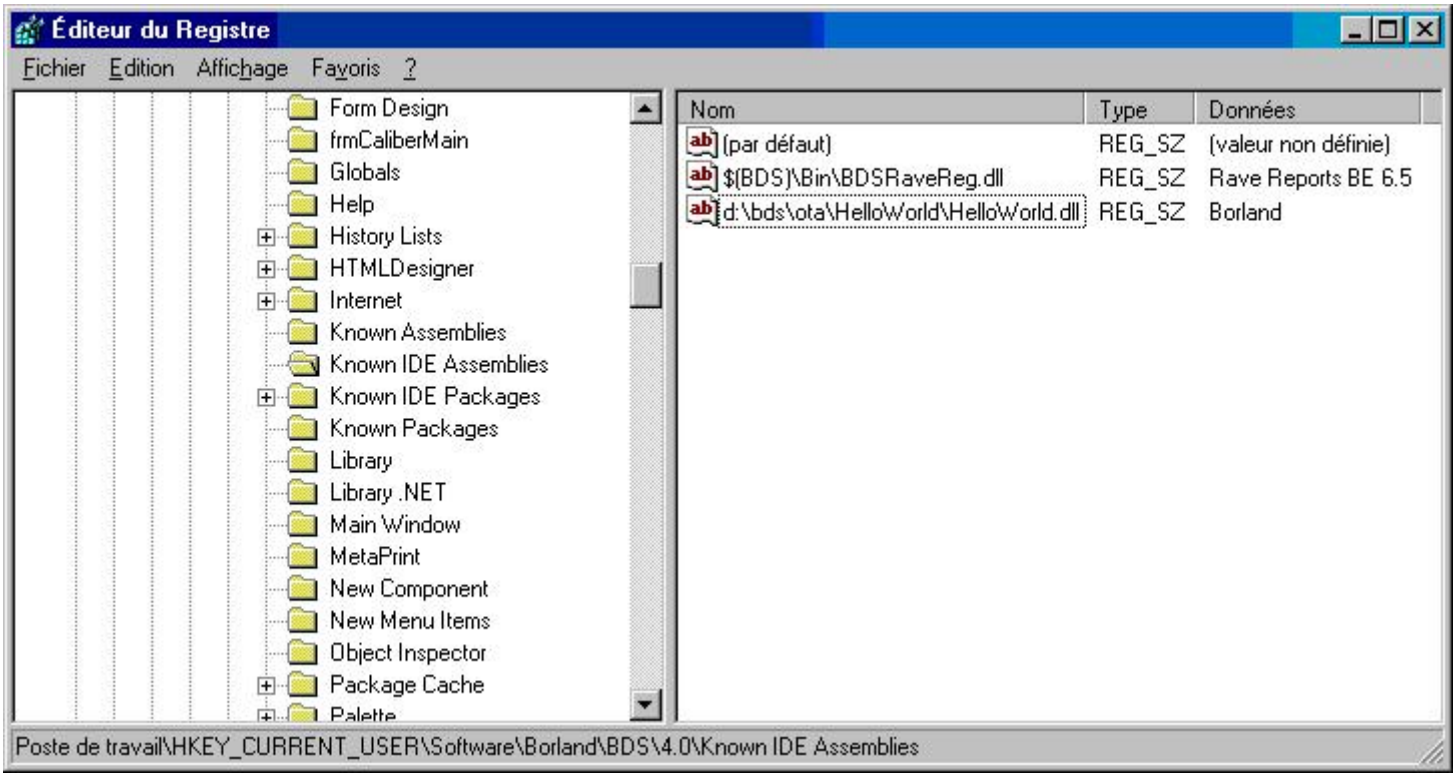
- Ajoutons une nouvelle classe avec une procédure bien spécifique :

```
[assembly: ComVisible(False)]
//[assembly: Guid('')]
//[assembly: TypeLibVersion(1, 0)]

type
  TMyFirstAddIn = class
  public
    class procedure IDERegister; static;
  end;

{ TMyFirstAddIn }

class procedure TMyFirstAddIn.IDERegister;
begin
  MessageBox.Show('Hello World !');
end;

begin
end.
```

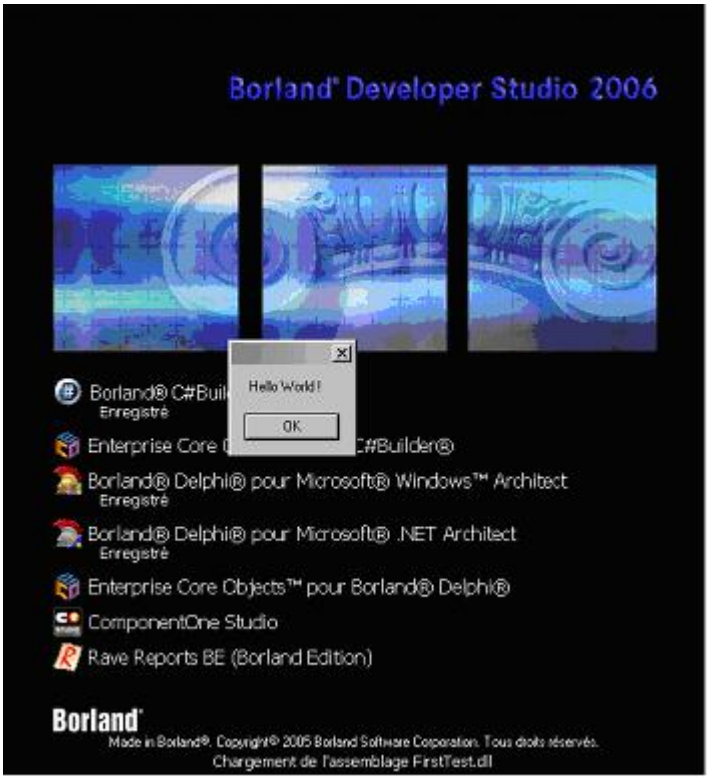
- Les assemblages .NET publie leur structure ce qui permet aux autres applications de facilement découvrir les fonctions et procédures qu'ils contiennent. BDS va se livrer à l'introspection de ses extensions pour en rechercher une procédure IDERegister, puis l'exécuter lorsqu'il l'aura trouvée.
- La classe MessageBox utilisée est contenue dans l'espace de noms : System.Windows.Forms (que l'on doit donc ajouter au uses de la bibliothèque).
- On peut ensuite sauvegarder son code et compiler son projet.

Maintenant, il faut que BDS charge cette nouvelle extension lorsqu'il se lance. Pour cela, il faut ajouter une nouvelle clé dans la base de registre :

- Lancez l'Editeur du Registre :
  - Menu Démarrer/Exécuter... (ou + R)
  - Tapez regedit puis cliquez sur OK
- Parcourez l'arborescence pour aller sur la clé : HKEY_CURRENT_USER\Software\Borland\BDS\4.0\Known IDE Assemblies (4.0 correspond à BDS 2006, 3.0 pour 2005, ...)
- Créez une nouvelle valeur chaîne et donnez-lui comme nom le chemin complet pour accéder à la DLL que nous venons de compiler. La valeur de la donnée par elle-même est sans importance mais ne doit pas être vide.



- Quittez BDS et relancez-le : Lors du chargement, le message Hello World ! apparaît...



Si vous souhaitez télécharger les fichiers du projet, vous pouvez le faire à partir de ce lien : http://borlandfrance.online.fr/delphi/bds_ota_helloworld.zip

Pour visualiser en ligne la vidéo de la création de ce projet, vous pouvez aller à cette adresse : http://borlandfrance.online.fr/delphi/

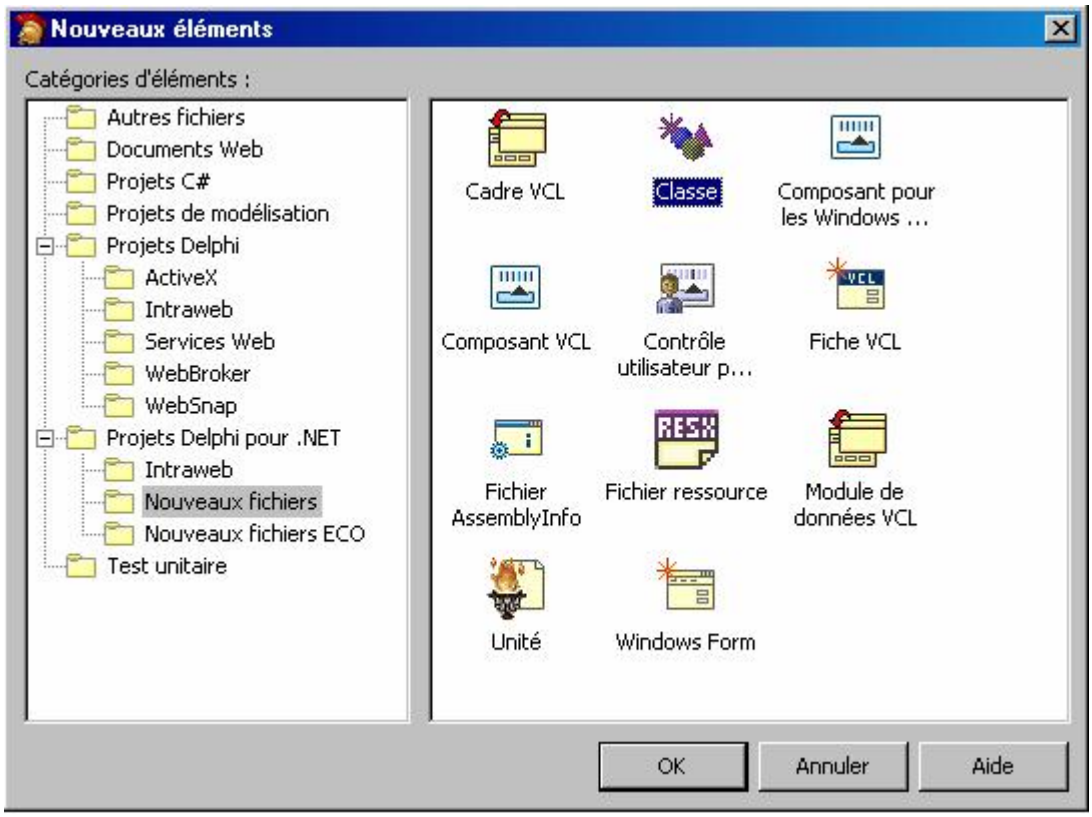
Nous avons ainsi réussi à créer notre première extension. Maintenant, nous allons essayer d'en faire une à peine plus complexe : La même mais où le Hello World sera affiché à partir d'un menu.

II - Créer un nouvel élément de menu dans l'EDI

Nous allons donc créer une nouvelle bibliothèque comme nous l'avons précédemment fait.

Nous lui ajoutons également la référence à l'assemblage Borland.Studio.ToolsAPI.

Puis nous créons une nouvelle classe.



Cette classe va contenir notre nouvelle extension. Elle hérite directement d'un TObject et implémente l'interface de l'OTA IOTAMenuWizard qui impose des propriétés permettant de récupérer le texte du menu, son nom, ...

Pour cela, on va modifier le code généré par BDS.

Pour commencer, on va utiliser les espaces de noms suivants :

```
uses
  Borland.Studio.ToolsAPI,
  System.Windows.Forms;
```

Puis faire la déclaration ainsi :

```
type
  TExtensionSimpleOTA = class(TObject, IOTAMenuWizard)
  strict private
    procedure CreateWizardService;
```

```
    private
      { Déclarations privées }
    public
      constructor Create;
      procedure   Execute;
      procedure   Destroyed;

      class procedure IDERegister; static;

      function FGetIDString : String;
      function FGetName     : String;
      function FGetMenuText : String;
      function FGetEnabled  : Boolean;
      function FGetChecked  : Boolean;

      property IDString     : String read FGetIDString;
      property Name         : String  read FGetName;
      property MenuText     : String  read FGetMenuText;
      property Enabled      : Boolean read FGetEnabled;
      property Checked      : Boolean read FGetChecked;
    end;

  implementation
```

On remarque que l'on retrouve comme dans le premier exemple la procédure qui permet à BDS d'enregistrer l'extension, IDERegister. Celle-ci va créer une instance de l'extension.

Dans le constructor de l'extension, on va faire appel à la procédure qui permet d'ajouter un nouvel expert à l'environnement, qui dans notre cas, sera un menu supplémentaire.

La procédure Execute sera quant à elle appelée lorsque l'on cliquera sur le menu et devra donc afficher le message Hello World.

L'implémentation ressemblera donc à ceci :

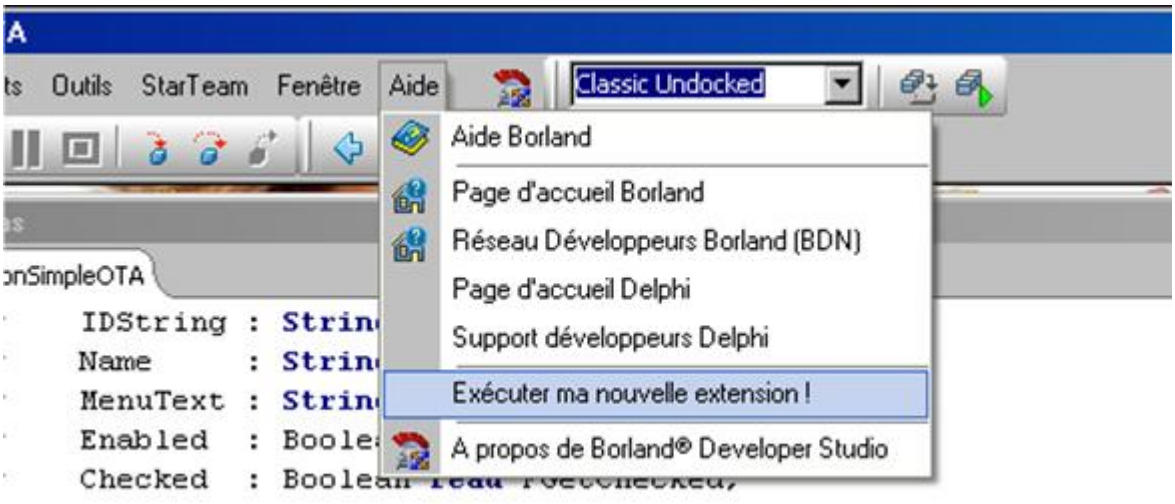
```
  implementation

  class procedure TExtensionSimpleOTA.IDERegister;
  begin
    TExtensionSimpleOTA.Create;
  end;

  constructor TExtensionSimpleOTA.Create;
  begin
    inherited Create;
    // TODO: Ajouter ici le code du constructeur
    CreateWizardService;
  end;

  procedure TExtensionSimpleOTA.CreateWizardService;
  var
    WizardService: IOTAWizardService;
  begin
    WizardService := BorlandIDE.GetService(typeof(IOTAWizardService)) as IOTAWizardService;
    WizardService.AddWizard(Self);
  end;

  procedure TExtensionSimpleOTA.Execute;
  begin
    MessageBox.Show('Hello World à partir du menu que l''on a créé !');
  end;

  procedure TExtensionSimpleOTA.Destroyed;
```

```
begin
  // Rien à faire : Le ramasse-miette s'en chargera...
end;

function TExtensionSimpleOTA.FGetChecked: Boolean;
begin
  Result := False;
end;

function TExtensionSimpleOTA.FGetEnabled: Boolean;
begin
  Result := True;
end;

function TExtensionSimpleOTA.FGetIDString: String;
begin
  Result := 'ExtensionSimpleOTA';
end;

function TExtensionSimpleOTA.FGetMenuText: String;
begin
  Result := 'Exécuter ma nouvelle extension !';
end;

function TExtensionSimpleOTA.FGetName: String;
begin
  Result := 'Extension simple via OTA';
end;

end.
```

Ensuite, il suffit de sauvegarder et de compiler son projet.

Puis on ajoute dans la base de registre l'appel à notre nouvelle extension (DLL).

On obtient ainsi un nouveau menu :



menu nous affiche effectivement le message que nous avons codé :

Si vous souhaitez télécharger les fichiers du projet, vous pouvez le faire à partir de ce lien : http://borlandfrance.online.fr/delphi/bds_ota_newItemMenu.zip

Maintenant, lors du chargement de BDS, vous avez remarqué que certaines extensions affichent un logo lorsqu'elles-mêmes sont chargées. Nous allons maintenant voir comment procéder pour faire la même chose.

III - Affichage de l'extension dans l'écran de démarrage de l'EDI (Splash screen)

Tout d'abord, nous allons avoir besoin d'une ressource image. Pour la créer, nous allons utiliser l'application Démo fournie avec BDS, ResXBuilder qui se trouve dans le répertoire : <BDS>\Demos\Delphi.Net\WinForms\ResXBuilder\

L'image doit faire une taille de 24x24.

Une fois le projet ResXBuilder ouvert, nous créons un nouveau fichier :

- Menu File/New

Puis nous ajoutons une ressource de type Bitmap :

- Menu Resources/Add/Bitmap...
- Choisir le fichier voulu et cliquer sur Ouvrir

Puis nous sauvegardons notre fichier :

- Menu File/Save

Ensuite, nous ouvrons notre projet précédent pour le modifier afin qu'il utilise notre logo fraîchement créé.

Nous ajoutons le logo à notre projet :

- Menu Projet/Ajouter au projet... (ou Maj + F11)
- Nous choisissons le type de fichier Ressources (*.resx)
- Nous cliquons sur le fichier créé avec le projet ResXBuilder et sur le bouton Ouvrir.

Après, nous modifions notre code pour qu'il utilise cette nouvelle ressource, en ajoutant une nouvelle procédure qui va être appelée à partir du constructor. Comme cette procédure va utiliser des espaces de noms particuliers, nous allons tout d'abord les ajouter :

```
uses
  Borland.Studio.ToolsAPI,
  System.Windows.Forms,
  System.Drawing,
  System.Resources,
  System.Reflection;
```

Puis, nous ajoutons la déclaration de la nouvelle procédure :

```
strict private
  procedure CreateWizardService;
  procedure CreateSplashImage;
private
```

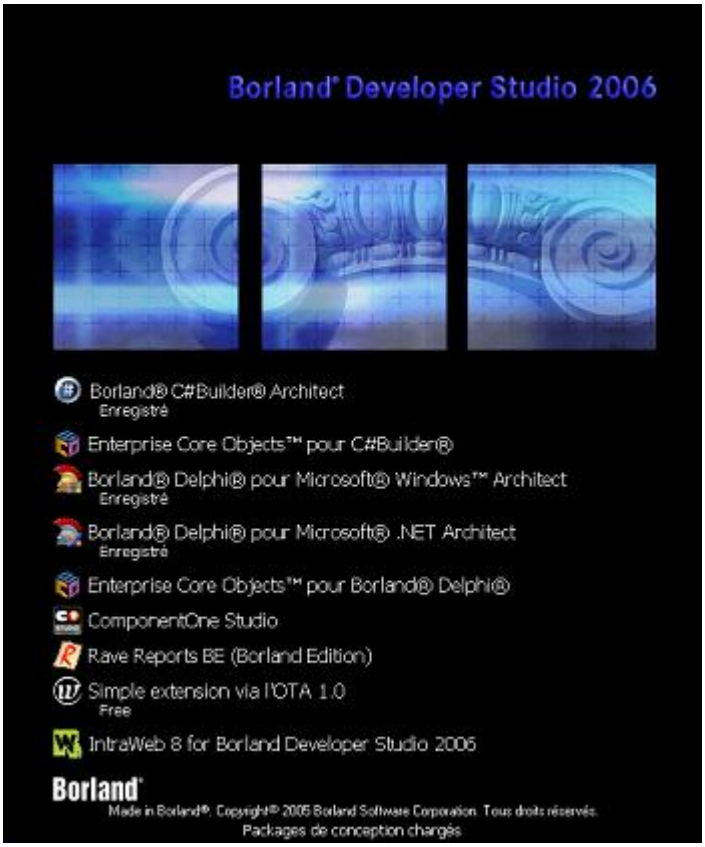
L'appel de cette procédure dans le constructor :

```
constructor TExtensionSimpleOTA.Create;
begin
  inherited Create;
  // TODO: Ajouter ici le code du constructeur
  CreateSplashImage;
  CreateWizardService;
end;
```

Et enfin, l'implémentation de la procédure :

```
procedure TExtensionSimpleOTA.CreateSplashImage;
var
  bmp                 : System.Drawing.Bitmap;
  SplashScreenService : IOTASplashScreenService;
  lAssembly           : System.Reflection.Assembly;
  leResourceManager   : System.Resources.ResourceManager;
begin
  // Récupération du service pour la splash
  SplashScreenService := BorlandIDE.GetService(typeof(IOTASplashScreenService)) as
 IOTASplashScreenService;

  // Récupération de l'assemblage courant
  lAssembly := GetType().Assembly;

  // Récupération du gestionnaire de ressources
  leResourceManager := System.Resources.ResourceManager.Create('whiler', lAssembly);

  // Récupération de l'image que nous avons créé (attention, sensible à la casse)
  bmp := System.Drawing.Bitmap(leResourceManager.GetObject('whiler.bmp'));

  // Ajout de l'image
  SplashScreenService.AddPluginBitmap('Simple extension via l''OTA', bmp.GetHbitmap, False, 'Free',
 '1.0');
end;
```

On remarque sur la capture ci-dessous l'emplacement des textes des différents paramètres :



Splash screen

Si vous souhaitez télécharger les fichiers du projet, vous pouvez le faire à partir de ce lien : http://borlandfrance.online.fr/delphi/bds_ota_splashScreen.zip

IV - Affichage de l'extension dans la fenêtre A propos de...

L'architecture ouverte que nous sommes en train d'utiliser a été conçue pour permettre à des éditeurs tiers de créer des extensions. De la même façon que nous avons ajouté des informations sur l'écran de démarrage (splash screen), nous allons maintenant ajouter des informations dans la fenêtre A propos de... de BDS.

Pour cela nous allons faire presque la même chose que pour la splash. L'image utilisée peut faire une taille de 32x32.

Nous créons une nouvelle procédure CreateAboutInfos que nous appelons à partir du constructor et que nous implémentons :

Déclaration de la nouvelle procédure :

```
  strict private
    procedure CreateWizardService;
    procedure CreateSplashImage;
    procedure CreateAboutInfos;
  private
```

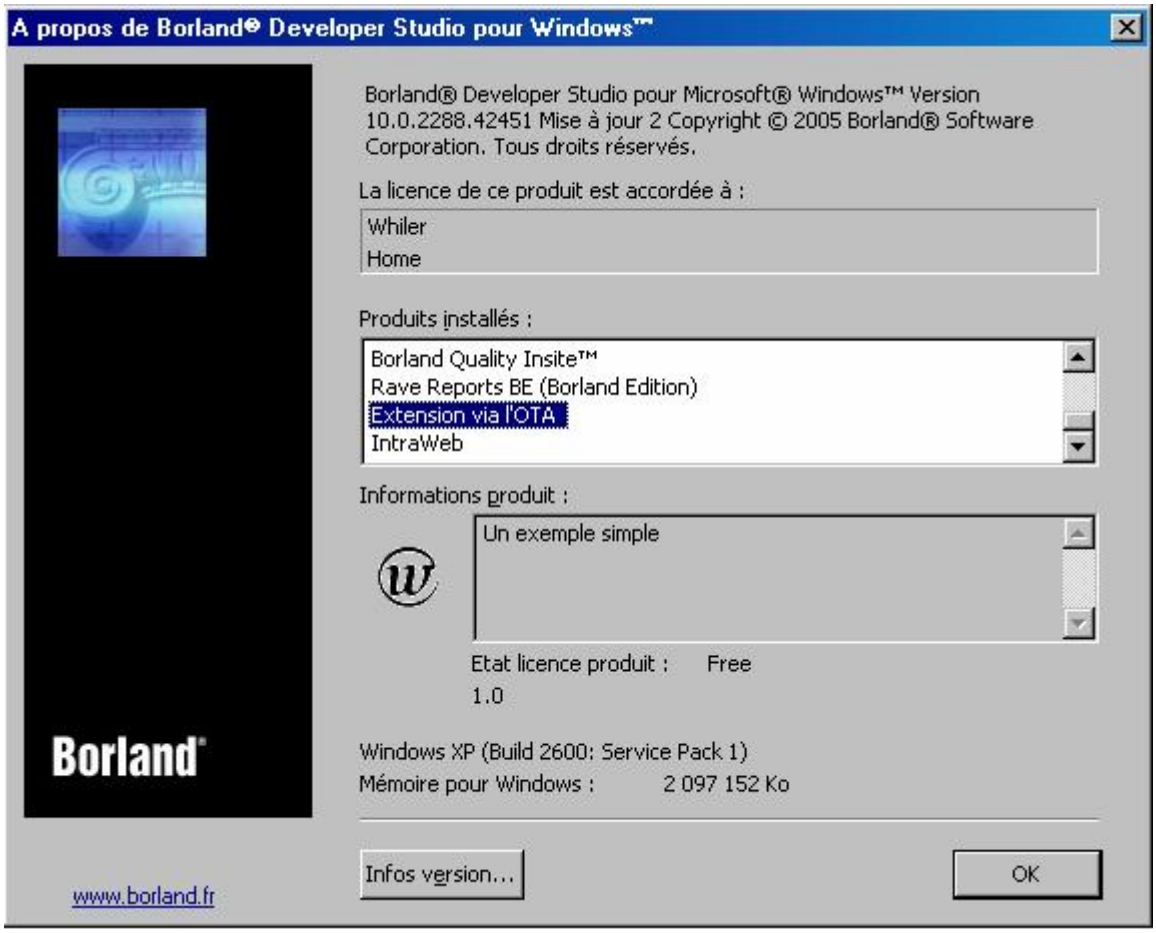
L'appel de cette procédure dans le constructor :

```
  constructor TExtensionSimpleOTA.Create;
  begin
    inherited Create;
    // TODO: Ajouter ici le code du constructeur
    CreateSplashImage;
    CreateAboutInfos;
    CreateWizardService;
  end;
```

Et enfin, l'implémentation de la procédure :

```
  procedure TExtensionSimpleOTA.CreateAboutInfos;
  var
    bmp                 : System.Drawing.Bitmap;
    AboutBoxService     : IOTAAboutBoxService;
    lAssembly           : System.Reflection.Assembly;
    leResourceManager   : System.Resources.ResourceManager;
  begin
    // Récupération du service pour le About
    AboutBoxService:= BorlandIDE.GetService(typeof(IOTAAboutBoxService)) as IOTAAboutBoxService;

    // Récupération de l'assemblage courant
    lAssembly := GetType().Assembly;

    // Récupération du gestionnaire de ressources
    leResourceManager := System.Resources.ResourceManager.Create('whiler', lAssembly);

    // Récupération de l'image que nous avons créé (attention, sensible à la casse)
    bmp := System.Drawing.Bitmap(leResourceManager.GetObject('about.bmp'));
    // On met le fond de l'image transparent
    bmp.MakeTransparent;

    // Ajout de l'image
    AboutBoxService.AddPluginInfo('Extension via l''OTA', 'Un exemple simple', bmp.GetHbitmap, False,
   'Free', '1.0');
  end;
```

On obtient ainsi :



Si vous souhaitez télécharger les fichiers du projet, vous pouvez le faire à partir de ce lien : http://borlandfrance.online.fr/delphi/bds_ota_about.zip

V - Création d'un élément de menu totalement personnalisé

Précédemment, nous avons vu comment ajouter un nouvel élément au menu d'Aide. Or, lorsque l'on crée une extension, on peut vouloir la voir apparaître dans un autre élément de la barre de menu. Nous allons maintenant voir comment placer notre menu ailleurs.

Pour cela, nous n'allons plus utiliser l'interface IOTAMenuWizard, mais l'interface IOTAMainMenuService.

Pour cela nous allons encore créer un nouveau projet de type bibliothèque.

Ajouter la référence sur l'API : Borland.Studio.ToolsAPI.dll

Ajouter au projet la ressource qui contient les images à afficher (logo, icône, ...)

Créer une nouvelle classe et sauvegarder tout le projet.

Dans la nouvelle unité que j'ai nommée ExtensionBody, nous allons remettre le code nécessaire que nous avons vu dans les exemples précédents (Attention aux retours à la ligne si vous copiez/collez tout le bloc) :

```
unit ExtensionBody;

interface

uses
  Borland.Studio.ToolsAPI,
  System.Windows.Forms,
  System.Drawing,
  System.Resources,
  System.Reflection;
type
  TExtensionOTA = class
  strict private
    procedure CreateSplashImage;
    procedure CreateAboutInfos;
  private
  public
    constructor Create;
    procedure Destroyed;
    class procedure IDERegister; static;
  end;

implementation

class procedure TExtensionOTA.IDERegister;
begin
  TExtensionOTA.Create;
end;

constructor TExtensionOTA.Create;
begin
  inherited Create;
  CreateSplashImage;
  CreateAboutInfos;
end;

procedure TExtensionOTA.CreateAboutInfos;
var
  bmp               : System.Drawing.Bitmap;
  AboutBoxService   : IOTAAboutBoxService;
  lAssembly         : System.Reflection.Assembly;
```

```
  leResourceManager : System.Resources.ResourceManager;
begin
  // Récupération de l'assemblage courant
  lAssembly := GetType().Assembly;

  // Récupération du gestionnaire de ressources {$R 'whiler.resources' 'whiler.resx'} le nom situé
avant .resources
  leResourceManager := System.Resources.ResourceManager.Create('whiler', lAssembly);

  // Récupération de l'image que nous avons créé (attention, sensible à la casse)
  bmp := System.Drawing.Bitmap(leResourceManager.GetObject('about.bmp'));
  bmp.MakeTransparent;

  // Récupération du service pour la fenêtre A propos de...
  AboutBoxService := BorlandIDE.GetService(typeof(IOTAAboutBoxService)) as IOTAAboutBoxService;

  // Ajout de l'image
  AboutBoxService.AddPluginInfo('Autre extension via l''OTA', 'Un autre exemple', bmp.GetHbitmap,
 True,
'Free', '1.0');
end;

procedure TExtensionOTA.CreateSplashImage;
var
  bmp                 : System.Drawing.Bitmap;
  SplashScreenService : IOTASplashScreenService;
  lAssembly           : System.Reflection.Assembly;
  leResourceManager   : System.Resources.ResourceManager;
begin
  // Récupération de l'assemblage courant
  lAssembly := GetType().Assembly;

  // Récupération du gestionnaire de ressources {$R 'whiler.resources' 'whiler.resx'} le nom situé
avant .resources
  leResourceManager := System.Resources.ResourceManager.Create('whiler', lAssembly);

  // Récupération de l'image que nous avons créé (attention, sensible à la casse)
  bmp := System.Drawing.Bitmap(leResourceManager.GetObject('whiler.bmp'));

  // Récupération du service pour la splash
  SplashScreenService := BorlandIDE.GetService(typeof(IOTASplashScreenService)) as IOTASplashScreen
Service;

  // Ajout de l'image
  SplashScreenService.AddPluginBitmap('Autre extension via l''OTA', bmp.GetHbitmap, True, 'Free',
'1.0');
end;

procedure TExtensionOTA.Destroyed;
begin
  // Rien à faire : Le ramasse-miette s'en chargera...
end;

end.
```

Nous pouvons normalement compiler le projet, sauf si des fautes de frappe se sont immiscées dans notre code.

Dans l'état actuel, cette nouvelle extension s'affiche lors du chargement de BDS et est également présente dans la fenêtre A Propos De... Mais elle ne fait rien de plus.

Nous allons donc créer une procédure permettant d'ajouter un menu qui lui-même permettra d'appeler une autre procédure qui exécutera le code que l'on souhaite.

Donc, nous déclarons une procédure CreateMainMenuItem ainsi :

```
    strict private
      procedure CreateSplashImage;
      procedure CreateAboutInfos;
      procedure CreateMainMenuItem;
    private
```

Dans son implémentation, nous allons récupérer l'icône à afficher (16x16), ajouter le nouvel élément de menu en spécifiant son emplacement par rapport à l'emplacement d'un élément de menu déjà existant.

```
procedure TExtensionOTA.CreateMainMenuItem;
var
  MainMenuService   : IOTAMainMenuService;
  menuItem          : IOTAMenuItem;
  bmp               : System.Drawing.Bitmap;
  lAssembly         : System.Reflection.Assembly;
  leResourceManager : System.Resources.ResourceManager;
begin
  // Récupération de l'assemblage courant
  lAssembly := GetType().Assembly;

  // Récupération du gestionnaire de ressources {$R 'whiler.resources' 'whiler.resx'} le nom situé
avant .resources
  leResourceManager := System.Resources.ResourceManager.Create('whiler', lAssembly);

  // Récupération de l'image que nous avons créé (attention, sensible à la casse)
  bmp := System.Drawing.Bitmap(leResourceManager.GetObject('menu.bmp'));
bmp.MakeTransparent;

  // Récupération du service pour la barre de menu
  MainMenuService := BorlandIDE.GetService(typeof(IOTAMainMenuService)) as IOTAMainMenuService;

  // Ajout de l'élément de menu
  menuItem := MainMenuService.AddMenuItem(MainMenuService.GetFirstMenuItem.Name,
 OTAMenuItemLocation.otamlBefore, 'Autre ExtensionOTA', 'Extension via l''OTA', bmp.GetHbitmap);
end;
```

Il faut juste faire appel à cette procédure dans le constructor, et nous aurons un nouvel élément de menu.

```
constructor TExtensionOTA.Create;
begin
  inherited Create;
  CreateSplashImage;
  CreateAboutInfos;
  CreateMainMenuItem;
end;
```

Par contre, nous n'avons pas encore spécifié ce que cet élément devait faire lorsque l'on clique dessus.

Pour cela, on crée une nouvelle procédure qui sera appelée lorsque l'on cliquera sur l'élément de menu :

Sa déclaration :

```
    public
      constructor Create;
      procedure Destroyed;
      class procedure IDERegister; static;
      procedure MenuItemExecuted(sender: TObject ; e: EventArgs);
    end;
```

Son implémentation :

```
procedure TExtensionOTA. MenuItemExecuted(sender: TObject ; e: EventArgs);
begin
  MessageBox.Show('Hello World !');
end;
```

Et on modifie la procédure CreateMainMenuItem pour que celle-ci spécifie l'utilisation de cette procédure :

```
  // Ajout de l'élément de menu
  menuItem := MainMenuService.AddMenuItem('ViewDebugItem', OTAMenuItemLocation.otamlBefore, 'Simple
ExtensionOTA', 'Extension via l''OTA', bmp.GetHbitmap);

  // Procédure à exécuter lorsque l'élément est cliqué
  Include(menuItem.Executed, MenuItemExecuted);
end;
```

Et puisque l'extension que vous allez créer sera très utile, vous allez vouloir lui affecter un raccourci clavier. Pour l'ajouter, il suffit simplement de modifier une nouvelle fois la procédure CreateMainMenuItem pour lui ajouter le raccourci choisi :

```
  // Ajout de l'élément de menu
  menuItem := MainMenuService.AddMenuItem('ViewDebugItem', OTAMenuItemLocation.otamlBefore, 'Simple
ExtensionOTA', 'Extension via l''OTA', bmp.GetHbitmap);

  // Ajout du raccourci clavier
  menuItem.Shortcut := Shift or Control or Alt or ord('W');

  // Procédure à exécuter lorsque l'élément est cliqué
  Include(menuItem.Executed, MenuItemExecuted);
end;
```

Où, shift, control et alt sont des constantes avec les valeurs suivantes :

```
const
  Shift   = $2000;
  Control = $4000;
  Alt     = $8000;
```

On peut ensuite compiler son projet, ajouter le chemin de la DLL dans la base de registre et relancer son BDS pour voir et tester son nouveau menu :

Si vous souhaitez télécharger les fichiers du projet, vous pouvez le faire à partir de ce lien : http://borlandfrance.online.fr/delphi/bds_ota_customItemMenu.zip

VI - Utilisation de la fenêtre de messages de BDS

Dans notre exemple, nous avons inséré notre menu en première position. Pour le placer ailleurs, il faut connaître le nom des autres éléments déjà existant.

Nous allons modifier notre code pour avoir ces noms et non plus un simple Hello World !

Nous commençons par ajouter trois unités que nous allons utiliser :

```
uses
  Borland.Studio.ToolsAPI,
  System.Windows.Forms,
  System.Drawing,
  System.Resources,
  System.Reflection,
  Borland.Vcl.Classes,
  Borland.Vcl.StrUtils,
  Borland.Vcl.Clipbrd;
```

Classes pour stocker les différents menus dans un TStringList, StrUtils pour gérer mon indentation par niveau et Clipbrd pour stocker le résultat dans le presse-papier.

Ensuite, nous créons une fonction qui parcourra récursivement les différents niveaux.

Sa déclaration :

```
type
  TExtensionOTA = class
  strict private
    procedure CreateSplashImage;
    procedure CreateAboutInfos;
  private
    function GetItems(root: IOTAMenuItem; level: Integer): TStringList;
  public
```

Son implémentation :

```
function TExtensionOTA.GetItems(root: IOTAMenuItem; level: Integer): TStringList;
var
  nextMenuItem : IOTAMenuItem;
  loop, max    : Integer;
begin
  Result := TStringList.Create;

  Result.Add(DupeString('- - ', level) + root.Name + ' | ' + root.Text);

  // Parcours des sous-menus...
  max := root.ChildCount - 1;
  for loop := 0 to max do
  begin
    result.AddStrings(GetItems(root.ChildMenuItem(loop), level+1));
  end;

  // Menu suivant...
  nextMenuItem := root.NextMenuItem;
  if (nextMenuItem <> nil) then
```

```
    begin
      result.AddStrings(GetItems(nextMenuItem, level));
    end;
  end;
```

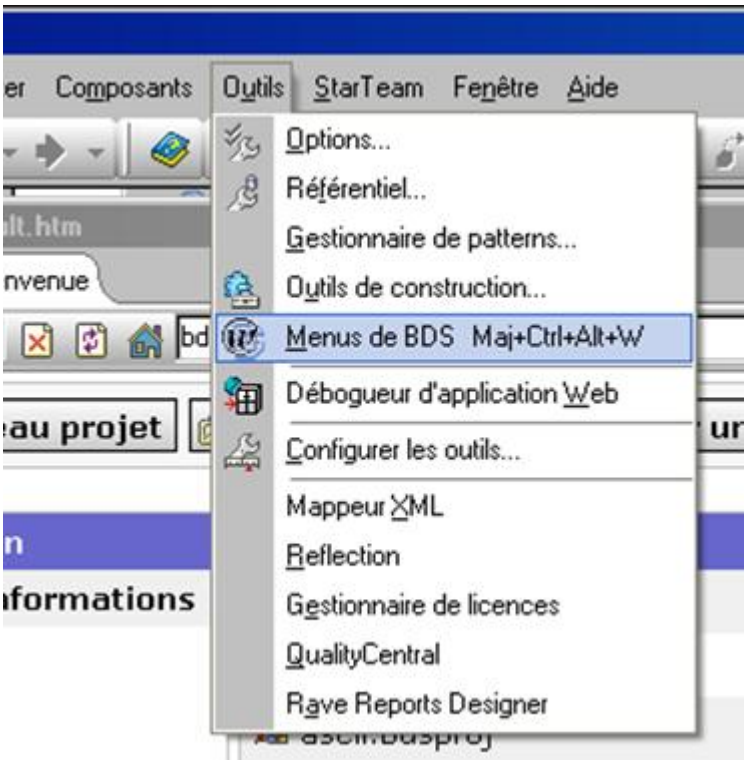
Puis nous modifions la procédure MenuItemExecuted pour récupérer la liste des menus :

```
procedure TExtensionOTA.MenuItemExecuted(sender: TObject; e: EventArgs);
var
  MainMenuService : IOTAMainMenuService;
  menuItems       : TStringList;
begin
  // Récupération du service pour la barre de menu
  MainMenuService := BorlandIDE.GetService(typeof(IOTAMainMenuService)) as IOTAMainMenuService;

  menuItems := GetItems(MainMenuService.GetFirstMenuItem, 0);

  Clipboard.AsText := menuItems.Text;
  MessageBox.Show('Menus copiés dans le presse-papier !');
end;
```

On peut ensuite compiler son projet, et tester son extension. Après avoir cliqué sur notre menu, on obtient la liste exhaustive des menus de BDS.

On peut donc déplacer son menu à un endroit plus approprié.

```
  // Ajout de l'élément de menu
  menuItem := MainMenuService.AddMenuItem('IDEToolsItem', OTAMenuItemLocation.otamlAfter,
 'MenusExtensionOTA', 'Menus de BDS', bmp.GetHbitmap);
```



La dernière étape que je vous propose dans ce document, est de ne plus utilisé le presse-papier pour stocker les menus mais de les afficher dans la fenêtre des messages.

Je vais supprimer l'unité qui gère le clipboard, et mettre à la place SysUtils pour pouvoir facilement transformer mes chaînes de caractères en nombre et réciproquement :

```
uses
  Borland.Studio.ToolsAPI,
  System.Windows.Forms,
  System.Drawing,
  System.Resources,
  System.Reflection,
  Borland.Vcl.Classes,
  Borland.Vcl.StrUtils,
  Borland.Vcl.SysUtils;
```

Pour cela, je vais modifier la fonction qui récupère les éléments de menu pour que je puisse avoir facilement le niveau et non pas une indentation :

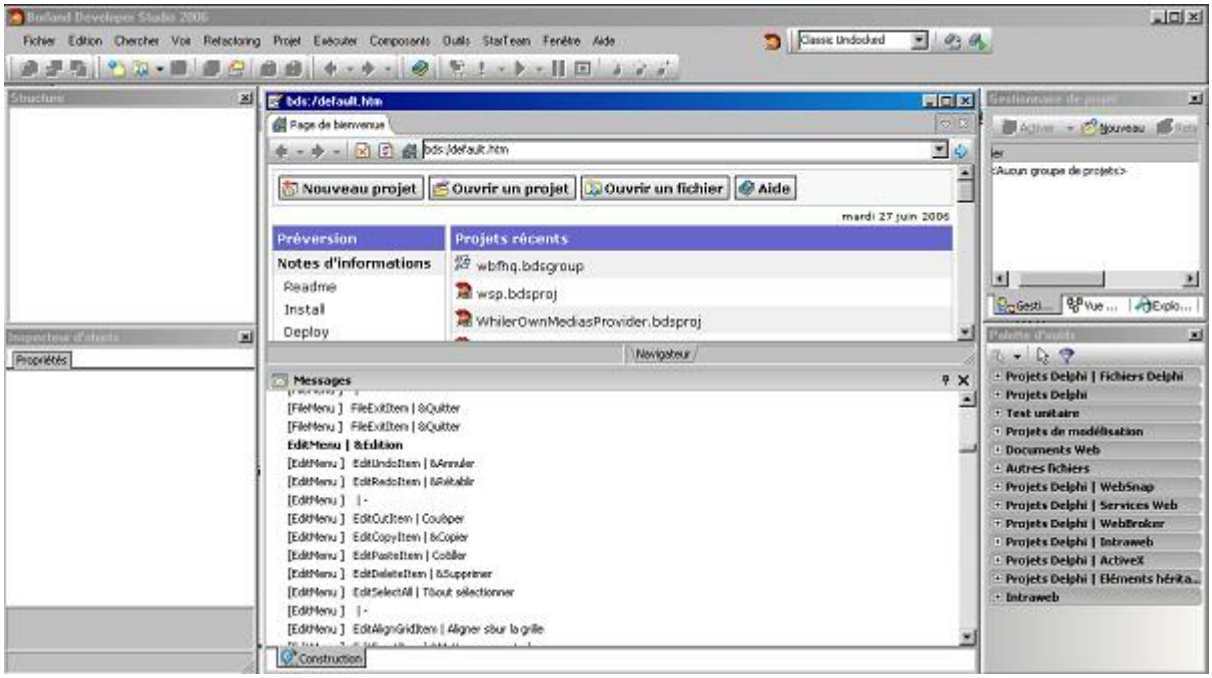
```
    Result.Add(IntToStr(level) + root.Name + ' | ' + root.Text);
```

Puis nous allons modifier la procédure MenuItemExecuted pour qu'elle utilise la fenêtre de messages de BDS et non plus le presse-papier :

```
procedure TExtensionOTA.MenuItemExecuted(sender: TObject; e: EventArgs);
var
  MainMenuService   : IOTAMainMenuService;
  menuItems         : TStringList;
  MessageService    : IOTAMessageService;
  oneMenu, display  : String;
  level             : Integer;
  mainTitleMenuName : String;
begin
  // Récupération du service pour la barre de menu
  MainMenuService := BorlandIDE.GetService(typeof(IOTAMainMenuService)) as IOTAMainMenuService;
  menuItems := GetItems(MainMenuService.GetFirstMenuItem, 0);

  // Récupère le service de message
  MessageService := BorlandIDE.GetService(typeof(IOTAMessageService)) as IOTAMessageService;

  // Supprime les messages précédents
  MessageService.ClearToolMessages;
  mainTitleMenuName := '';

  for oneMenu in menuItems do
  begin
    level   := StrToIntDef(oneMenu[1], 0);
    display := RightStr(oneMenu, Length(oneMenu) - 1);
    if (level = 0) then
    begin
      // Texte en caractères gras
      MessageService.AddTitleMessage(display);
mainTitleMenuName := LeftStr(display, Pos(' |', display));
    end
    else
    begin
      // Texte standard
      MessageService.AddToolMessage('', DupeString(' ', level) + display, mainTitleMenuName, 0, 0);
    end;
  end;
```

```
    // Affiche la vue des messages
    MessageService.ShowMessageView(nil);
  end;
```

Il suffit ensuite de recompiler, et nous obtenons ainsi la liste exhaustive des éléments du menu de BDS dans notre fenêtre de message.



Si vous souhaitez télécharger les fichiers du projet, vous pouvez le faire à partir de ce lien : http://borlandfrance.online.fr/delphi/bds_ota_messagePanel.zip

VII - Conclusion

J'espère avoir été suffisamment clair dans ce tutorial et qu'il vous aura permis de plus facilement aborder l'architecture qui vous permettra ainsi de créer vos propres extensions.

Si vous obtenez cette erreur lors de la compilation de votre projet, Erreur du Linker pendant l'émission des métadonnées (E2328), c'est que votre extension a été montée dans BDS et que celui-ci ne peut par conséquent pas recréer la DLL lors de la compilation.

Voici quelques liens qui m'ont été particulièrement utiles lors de la rédaction de cet article :

- **http://delphi.about.com/library/weekly/aa101204a.htm**
- **http://www.delphi3000.com/articles/article_3870.asp?SK=**
- **http://bdn1.borland.com/borcon2004/article/paper/0,1963,32119,00.html**
- **http://dn.codegear.com/article/30194**