Utilisation de AppExpert.

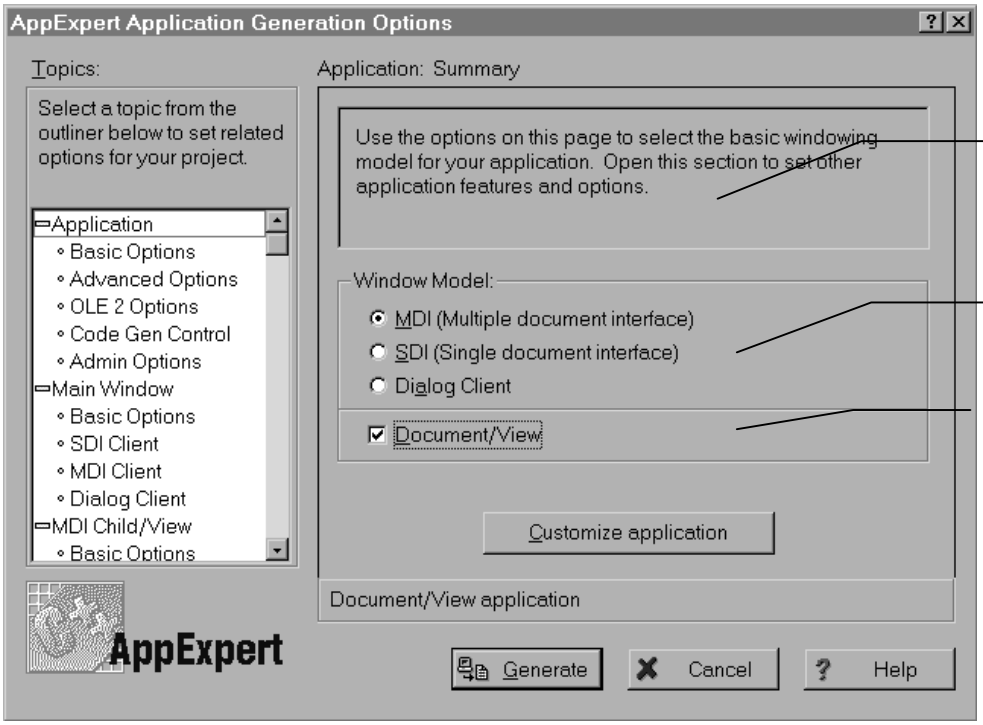
AppExpert est un outil de prototypage rapide d'applications efficace et pratique à utiliser. Son utilisation conjointe avec ClassExpert permet de gagner beaucoup de temps lors de la conception de nouvelles applications. Toutefois, sa fragilité le rend difficile à utiliser sur de grands projets. En outre, une fois les options spécifiées, il n'est pas possible de relancer AppExpert pour les modifier.

Insistons sur l'aspect didactique de AppExpert qui permet de générer des exemples très intéressants sur les fonctionnalités avancées de Windows que sont la gestion du modèle Document/Visualisation, les barres d'outil déplaçables, les lignes d'état, la gestion du glisser/déposer etc ...

1 Création d'une application avec AppExpert

Pour créer une application avec AppExpert, il faut choisir New–App Expert dans le menu File de Borland C++.

On se retrouve alors devant la boîte de dialogue principale d'AppExpert qui permet de choisir les grandes orientations de l'application que vous souhaitez créer. La figure suivante montre l'arborescence des options déployée ainsi que le premier écran d'options



Aide minimale sur les choix courants

Choix du type d'application

Sélection du modèle Document/Visualisation

Dans le cas d'une application MDI, il est fortement recommandé de cocher cette option.

Dans un premier temps, il vous est demandé de choisir le type de fenêtre principale d'application que vous désirez, vous avez le choix entre :

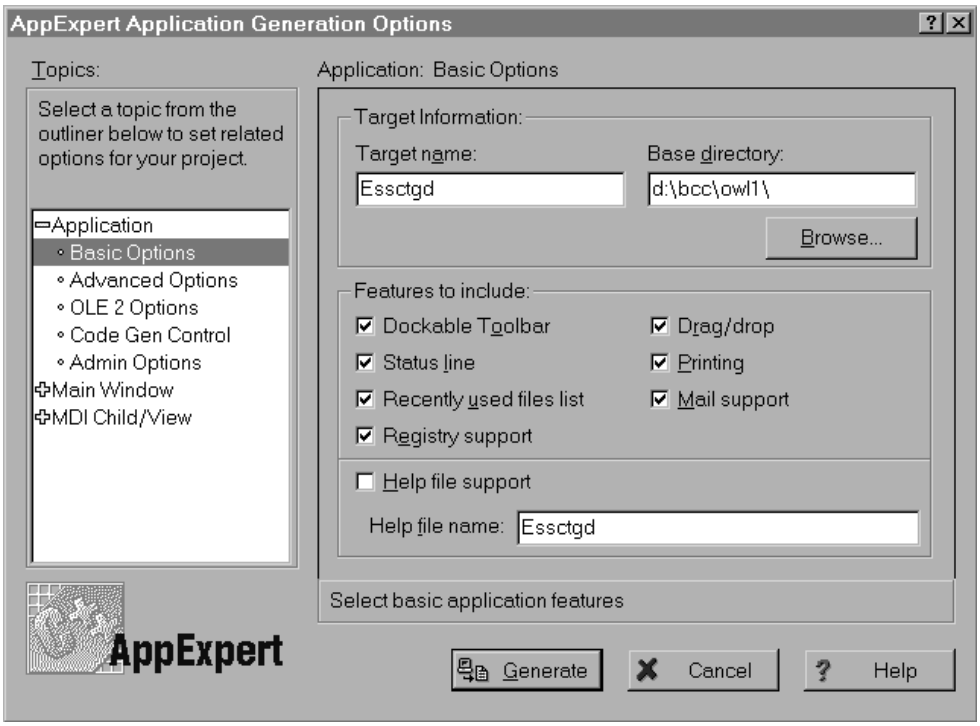
- Une application MDI genre Word ou Excel.
- Une application SDI où un seul document peut être ouvert à la fois.
- Une application dont la fenêtre principale est une boîte de dialogue.

Dans ce dernier cas, une fenêtre intermédiaire de type TFrameWindow est intercalée entre la classe application et la boîte de dialogue pour des raisons qui me sont inconnues. Néanmoins, une fois l'application finalisée, il est possible de faire sauter cette couche sans aucun difficulté et de connecter directement la boîte de dialogue à l'application

1.1 Les choix liés à l'application

1.1.1 Basic Options

Cette boîte fondamentale permet non seulement de spécifier le nom de l'exécutable ainsi que le répertoire de base de génération du code mais également les enrichissements de l'application comme le montre la recopie d'écran suivante :



Détaillons les plus complexes de ces options

Dockable Toolbar : Idéal pour générer une barre d'outils (Toolbar) que l'on peut déplacer ou transformer en palette flottante à volonté (Dockable). Je recommande de toujours cocher cette option car la barre par défaut est très facile à modifier. En outre, il suffit de mettre en commentaire 2 lignes pour déconnecter cette barre d'outils si on ne la désire pas alors qu'il est assez ardu d'en créer une de toutes pièces.

Status line : crée une ligne de status en bas de la fenêtre qui contient un gadget text ainsi qu'une série de gadgets permettant d'afficher l'état des touches du clavier.

Printing : rajoute des commandes de menu et, éventuellement des boutons dans la barre d'outils proposant à l'utilisateur de prévisualiser et d'imprimer son document.

Registry-support : cette commande, particulièrement dangereuse au demeurant ajoute à votre application la possibilité de modifier la base de registres de l'ordinateur hôte afin d'associer votre à une extension. Lorsque l'on sait que l'extension par défaut est .txt, on imagine aisément les conséquences d'une mauvaise utilisation de cette option. Aussi, je ne la recommande qu'aux utilisateurs avancés et lorsque l'on spécifie une extension de document non utilisée pour les documents traités par son application.

Mail support : permet d'envoyer par email le document courant après encodage du document.

Drag-drop : met en place un canevas général pour utiliser les facilités de glisser/déposer. Une fois de plus, le caractère didactique de cette option fait merveille.

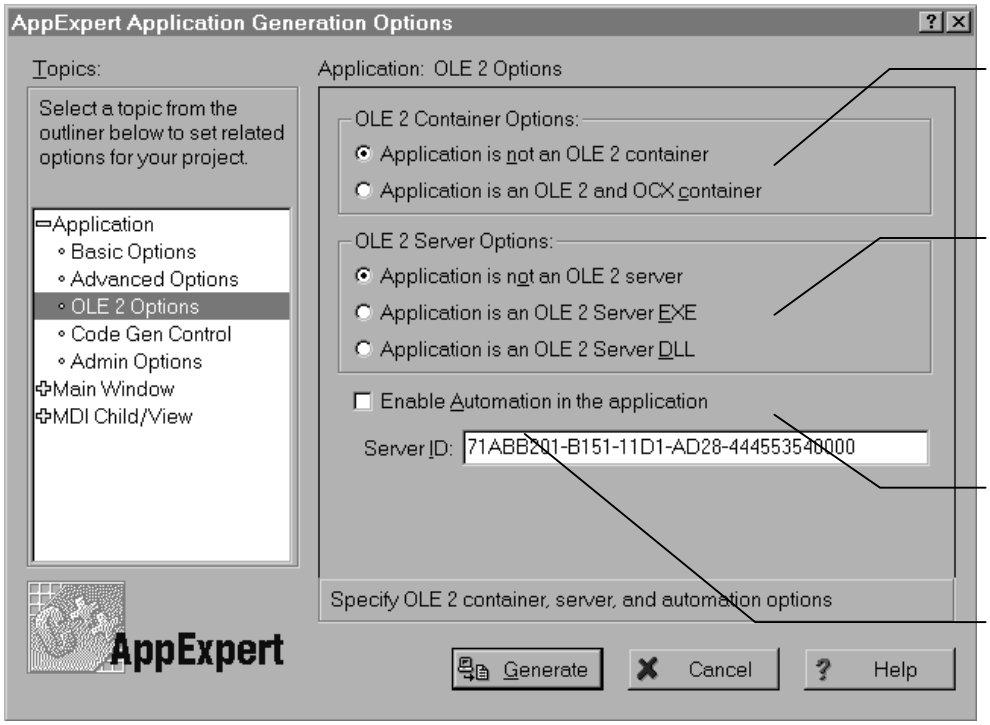
Bien entendu, pour les fonctionnalités telles que l'impression ou le support du glisser/déposer, la plupart du code reste à la charge du développeur, mais la mise en place des classes avec les prototypes des méthodes permet de considérablement simplifier la tâche de conception.

1.1.2 Advanced Options

Malgré son nom, cette barre d'options très simple ne propose que quelques options concernant l'aspect et les dimensions de la fenêtre d'application lors du lancement de l'exécutable. Nous ne les détaillerons pas.

1.1.3 OLE 2 Options

Ce volet permet de spécifier les options OLE de votre application. Les options sont détaillées en légende de la figure.



Indique que votre application est capable d'insérer des composants OLE

Indique que votre application est capable d'exporter des composants OLE vers d'autres applications, soit en mode EXE, soit en mode DLL

Indique que votre application exporte des procédures (automate)

Identificateur OLE (éventuel) de votre application

1.1.4 Code Gen Control

Permet de gérer la génération du code de AppExpert (utilisation des noms de fichiers longs, présence de commentaires automatiques, séparation de répertoires pour le code et les include, etc...)

1.1.5 Admin Options

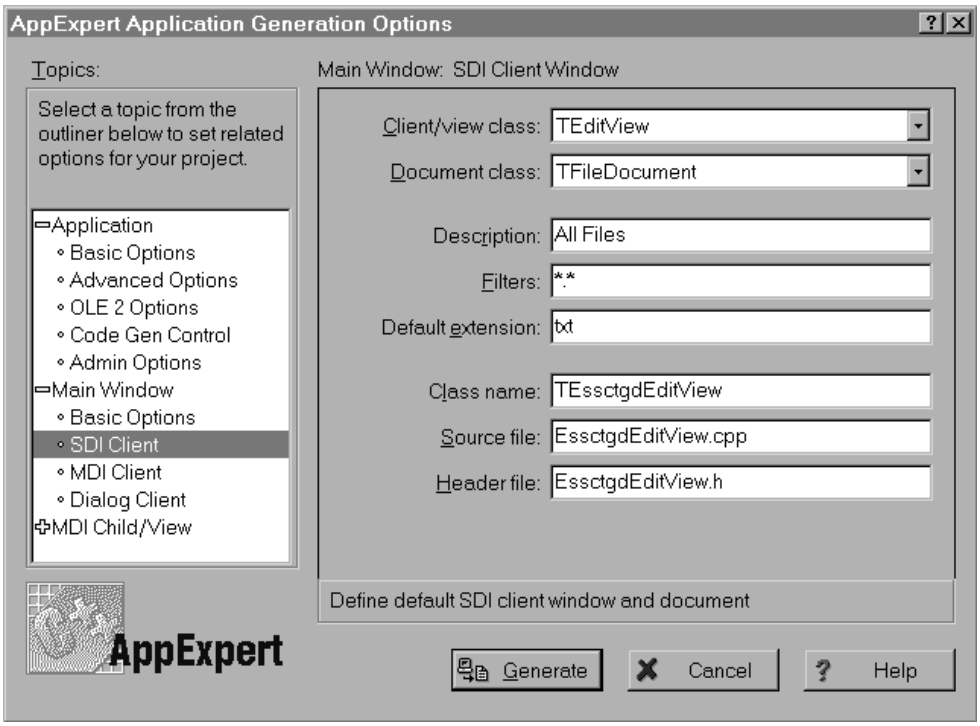
Ces options, très ludiques, sont liées à la boîte de dialogue « A propos » et permettent de spécifier le nom de l'entreprise éditant le logiciel, le numéro de version, etc.

1.2 Les options concernant la fenêtre principale d'application

En sus du choix du nom de la classe représentant la fenêtre principale, les différents volets permettent de choisir diverses options concernant la fenêtre principale. Certaines options peuvent ne pas avoir de sens dans votre cas courant. En effet, il n'est pas utile de s'intéresser à la rubrique DialogClient si l'on a choisi une application MDI.

Quels que soient les choix réalisés, il est déconseillé de toucher à quoi que ce soit dans la rubrique MDI Client.

Finalement, ces options ne sont intéressantes que dans le cas SDI. Le volet se présente alors comme suit :



La présentation exacte dépend de l'adoption ou non du modèle Document/Visualisation. Dans l'exemple ci-dessus, ce choix a été fait.

Il s'agit ici de choisir le nom des classes à utiliser ainsi que l'extension par défaut des documents utilisés. Attention, si vous avez choisi d'enregistrer votre application et s'il existe déjà une application associée à cette extension, cette dernière sera retirée de la base de registre au profit de l'application en cours de création.

Les options Description et Filters permettent de mettre en place les options de même nom des boîtes de dialogue standard d'ouverture et enregistrement de fichier de Windows 95.

Au cas ou vous ne suivriez pas le modèle Document/Visualisation, trois classes de base sont prévues pour la zone client de votre fenêtre :

- **TWindow** : fenêtre de base quelconque
- **TRichEdit** :votre application est alors pourvue dès le départ de fonctionnalités d'édition de texte non négligeables
- **TListBox** : la fenêtre principale se comporte comme une liste d'options

1.3 Options spéciales MDI

Les seules options susceptibles de poser problèmes sont celles concernant les fenêtres enfant du modèle MDI. Les options proposées sont similaires à celles régissant le fonctionnement de la fenêtre principale d'une application SDI (voir ci-dessus).

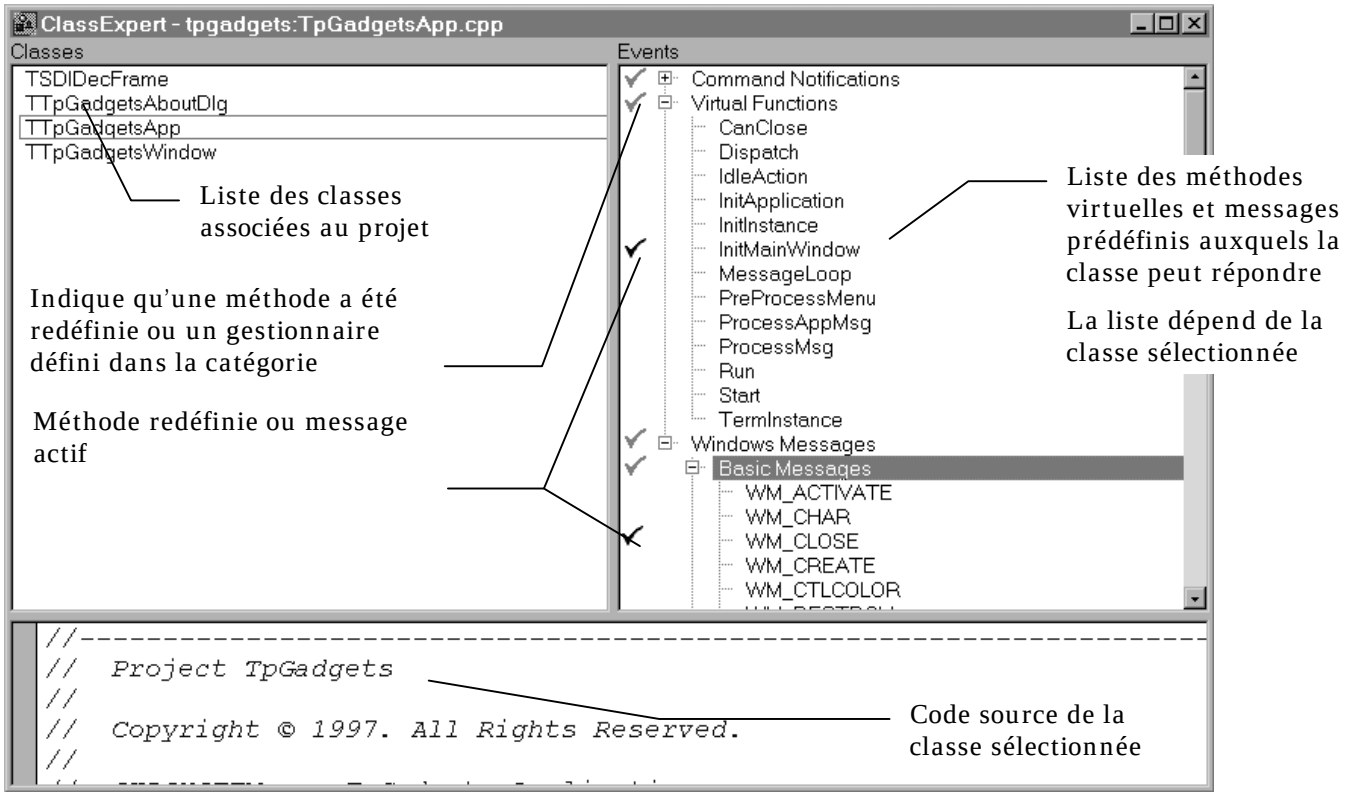
Au risque de se répéter, rappelons toutefois qu'il est fortement recommandé de suivre le modèle Document/Visualisation pour concevoir une application MDI.

2 Modification de l'application avec Class Expert

Lorsque l'on a créé une application avec AppExpert, l'action par défaut associée à une cible dans la fenêtre de projet n'est plus l'exécution (toujours accessible dans le menu Debug ou via le raccourci Ctrl-F9) mais l'appel de l'utilitaire ClassExpert qui permet d'interagir efficacement avec le code OWL en dépit d'une grande fragilité.

2.1 Présentation générale de ClassExpert

La fenêtre principale de ClassExpert se présente ainsi :



Le volet de gauche recense les classes OWL du projet. Les classes non OWL n'y sont jamais répertoriées.

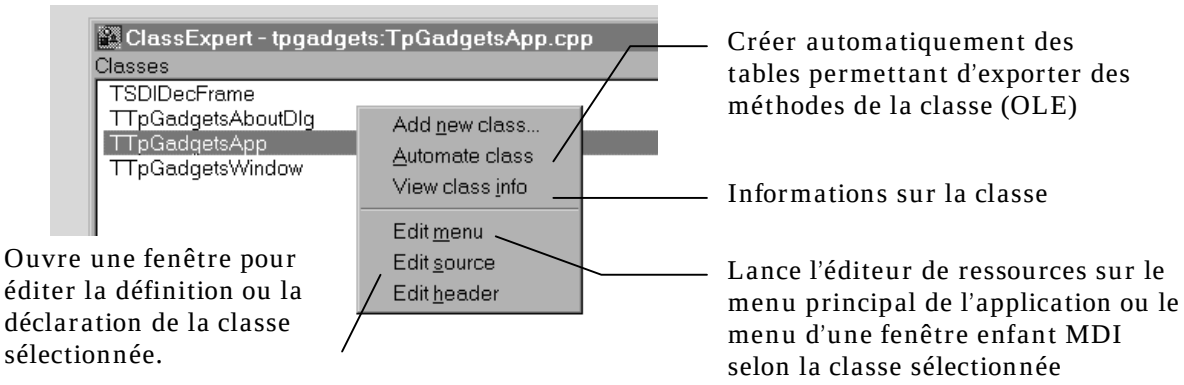
Le volet de droite liste les différentes possibilités de redéfinition proposées par la classe sélectionnée à gauche. On y retrouve les méthodes virtuelles prédéfinies ainsi que la liste des messages auxquels la classe peut répondre. Ceux-ci sont séparés en notifications de commande, en provenance, par exemple, d'un menu et en messages Windows, ces derniers étant regroupés par catégories.

2.2 Les commandes ClassExpert

Toutes les commandes de ClassExpert sont regroupées dans les menus rapides associés au bouton de droite.

Les différentes commandes permettent aussi bien de rajouter des classes dans le projet que des méthodes de gestion des événements de Windows ou même de redéfinir les méthodes virtuelles prédéfinies des classes d'OWL.

2.2.1 Manipulation de la liste des classes



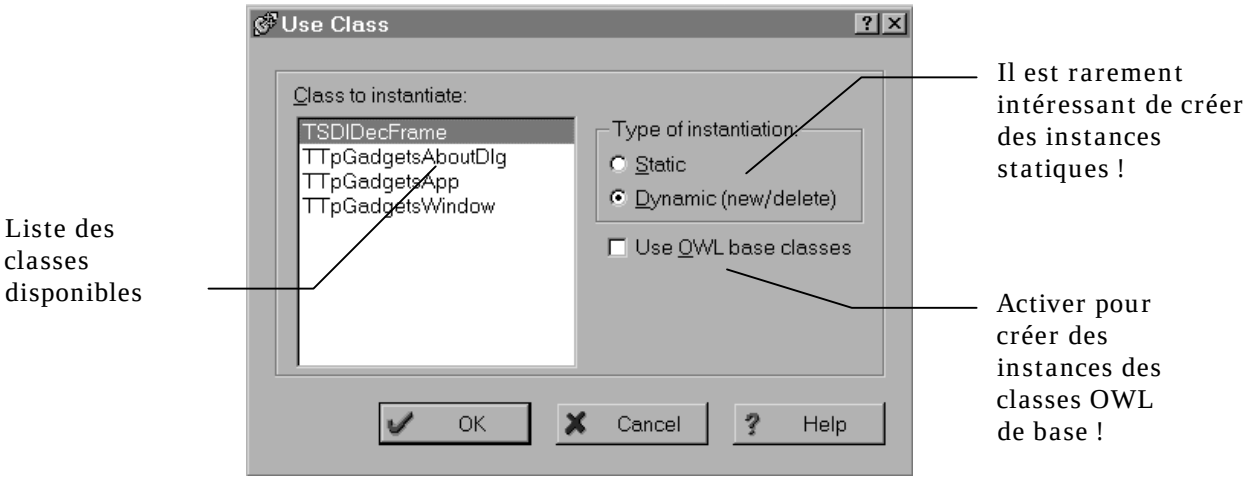
La commande la plus intéressante est « Add new class » qui permet d'ajouter une nouvelle classe au projet. La classe ajoutée dérive obligatoirement d'une classe OWL. Notez qu'il n'existe pas de commande permettant de supprimer une classe. Ceci doit être fait dans l'éditeur de projet. Il est alors nécessaire d'invoquer la commande « Rescan » afin de mettre à jour les informations AppExpert (dans le fichier *projet*.apx).

L'ajout d'une classe entraîne la création d'un squelette minimal de classe incluant un constructeur et un destructeur. C'est toujours la forme la plus complète du constructeur qui est fournie.

Le volet d'édition du code source ne peut contenir que du code source (.cpp). Il faut faire très attention à ne pas éditer en même temps le même fichier dans ce volet et une fenêtre d'édition normale de Borland C++ : les mises à jour risquent de ne pas se faire correctement. Le résultat en est habituellement une GPF (normal on est sous Windows ☺).

2.2.2 Le menu rapide du volet d'édition

En plus des commandes d'édition habituelles, le menu rapide du volet d'édition propose une commande nommée « Use class ». Cette dernière permet de créer une instance d'une classe OWL du projet ou même d'une classe OWL standard. La boîte de dialogue invoquée est la suivante :



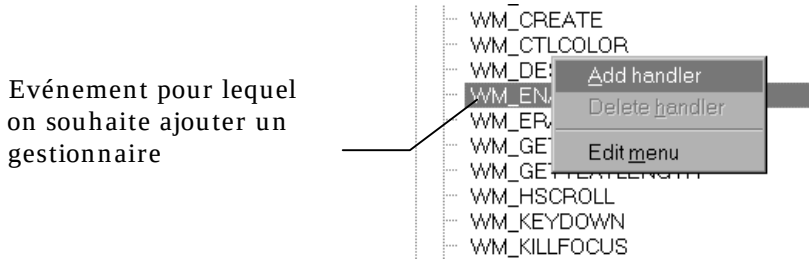
Le code généré est du type :

```
//  TTpGadgetsAboutDlg*   xTTpGadgetsAboutDlg = new TTpGadgetsAboutDlg(TWindow*
parent, TResId resId = IDD_ABOUT, TModule* module = 0);
```

Notons que les directives #include nécessaires sont automatiquement générées. Cette fonctionnalité est particulièrement intéressante lorsque l'on ne connaît pas par cœur la liste (souvent impressionnante) des paramètres d'un constructeur ou le fichier include associé à une classe.

2.2.3 Le menu rapide du volet des méthodes

C'est le menu qui permet de redéfinir les méthodes virtuelles prédéfinies des classes OWL ou d'ajouter des gestionnaires de menu. Pour rajouter un événement, il convient de sélectionner l'événement désiré dans la liste puis d'invoquer la commande « Add handler » comme indiqué par la figure ci-dessous.



Suit une boîte de dialogue qui invite l'utilisateur à saisir un nom pour la nouvelle procédure. La génération de code consiste à ajouter une méthode protected dans la classe et à fournir un squelette minimal, par exemple :

```
void TEssaiDlgClient::BNClicked()
{
  // INSERT>> Your code here.
}
```

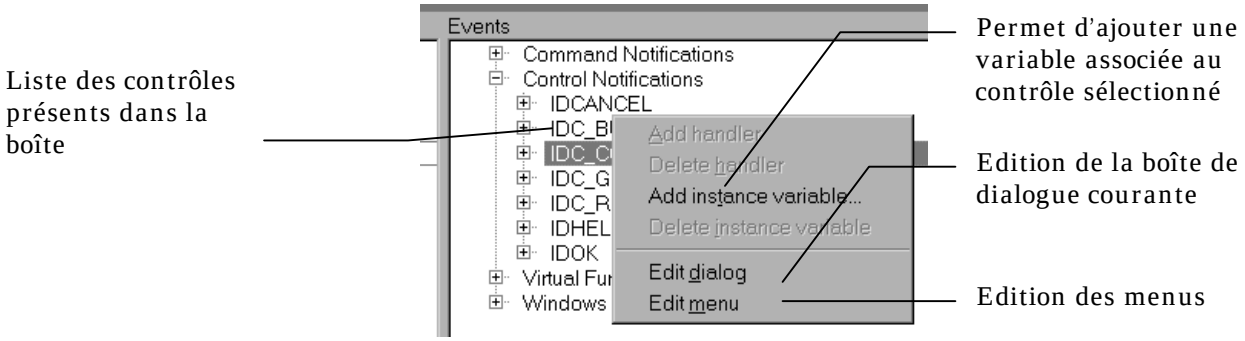
La même démarche est utilisée pour redéfinir les méthodes virtuelles. Par défaut, les méthodes surchargés commencent par appeler la méthode de la classe mère.

Ce même menu propose également de supprimer un gestionnaire existant. La méthode n'est pas supprimée de la classe, seule son entrée dans la RESPONSE_TABLE est désactivée.

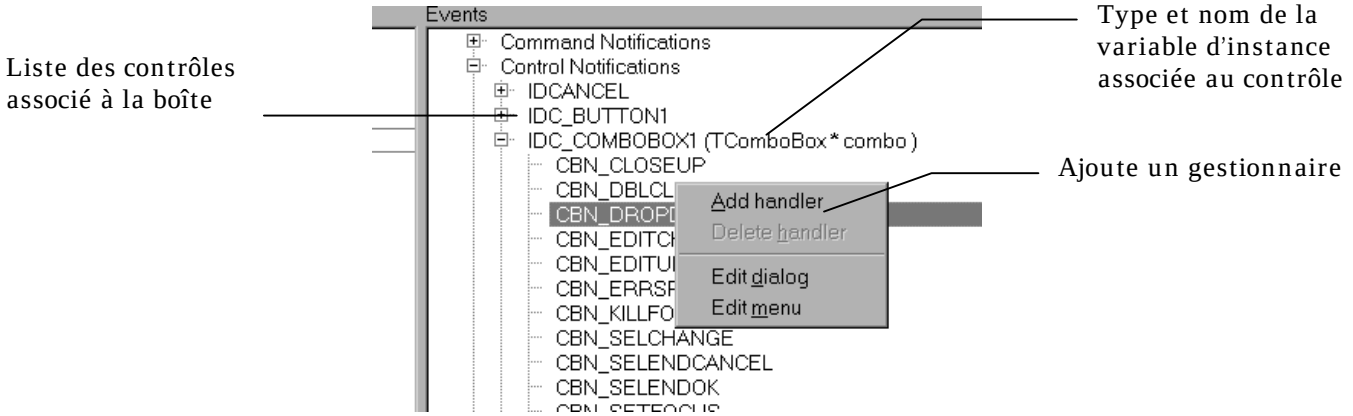
Dans le cas des boîtes de dialogue, une nouvelle catégorie nommée « Control notifications » est disponible. Son but est d'associer des méthodes aux événements générés par les différents contrôles présents dans la boîte et de gérer les variables d'instance de la classe boîte qui y sont associés.

La rubrique « Control Notifications » est représentée sous la forme de la liste des contrôles qui peuplent la boîte de dialogue (voir figure ci-dessous). Pour chaque contrôle il est possible d'ajouter (puis de retirer) une variable d'instance. Son nom et son type apparaissent alors à droite de l'ID de ressource. L'instanciation est réalisée automatiquement dans le constructeur de la boîte de dialogue. Il est beaucoup plus sur de réaliser cette opération en utilisant ClassExpert qu'à la main. En effet, ClassExpert associera toujours le bon ID de ressource à la variable adéquate.

Notons toutefois que certaines ressources ne sont pas autant initialisés qu'on le souhaiterait. En effet, si certains boutons radio sont regroupés dans une boite de groupe (GroupBox) leur construction ne prend pas en compte cet aspect.



Il est également possible de traiter directement les événements de notification des contrôles. Cette opération est totalement indépendante de l'association à des variables d'instance. Par exemple, il est très fréquent de gérer l'événement BN_CLICKED associé à un bouton sans pour autant lui associer une variable. La commande « Add handler » du menu de droite invoqué sur un événement permet de créer un tel gestionnaire comme le montre la figure suivante. Le code du gestionnaire est semblable à celui généré pour tout autre événement.



Pour conclure notons que les événements de type EV_CHILDxxx ne sont pas gérables directement par ClassExpert, il faut donc s'en occuper à la main. En outre, la RESPONSE_TABLE d'une classe n'est créée que si des gestionnaires ont été spécifiés. Aussi, si vous ne désirez traiter que des événements non compris par ClassExpert, il faudrait, en toute rigueur, déclarer et définir la table de réponse à la main. Il est possible de pallier à ce facheux inconvénient en demandant un gestionnaire d'événement que vous désactivez immédiatement après ce qui aura pour effet de créer la table des réponses. Vous pourrez alors ajouter tout ce que vous voudrez dedans en toute impunité et sans affaiblir vos chances de déclencher une GPF bien méritée !

2.3 Les dangers de ClassExpert

ClassExpert est un outil fort sympathique mais néanmoins capricieux. En effet, certaines modifications sont à proscrire. Par exemple, il ne faut pas rajouter à la main des identificateurs de ressource dans le fichier .rh mais toujours passer par l'éditeur de ressources.

En outre, les commentaires spéciaux de ClassExpert sont à respecter absolument si l'on veut qu'il puisse s'y retrouver. En effet, ils délimitent les zones de déclaration et de définition des méthodes virtuelles ou des gestionnaires de messages. Ils sont de la forme :

```
//{{TEssaiDlgClientRSP_TBL_BEGIN}}
```

Si le projet ClassExpert devient corrompu, il est toutefois possible de continuer à travailler indépendamment en le détruisant puis en bâtissant un projet normal à la place.