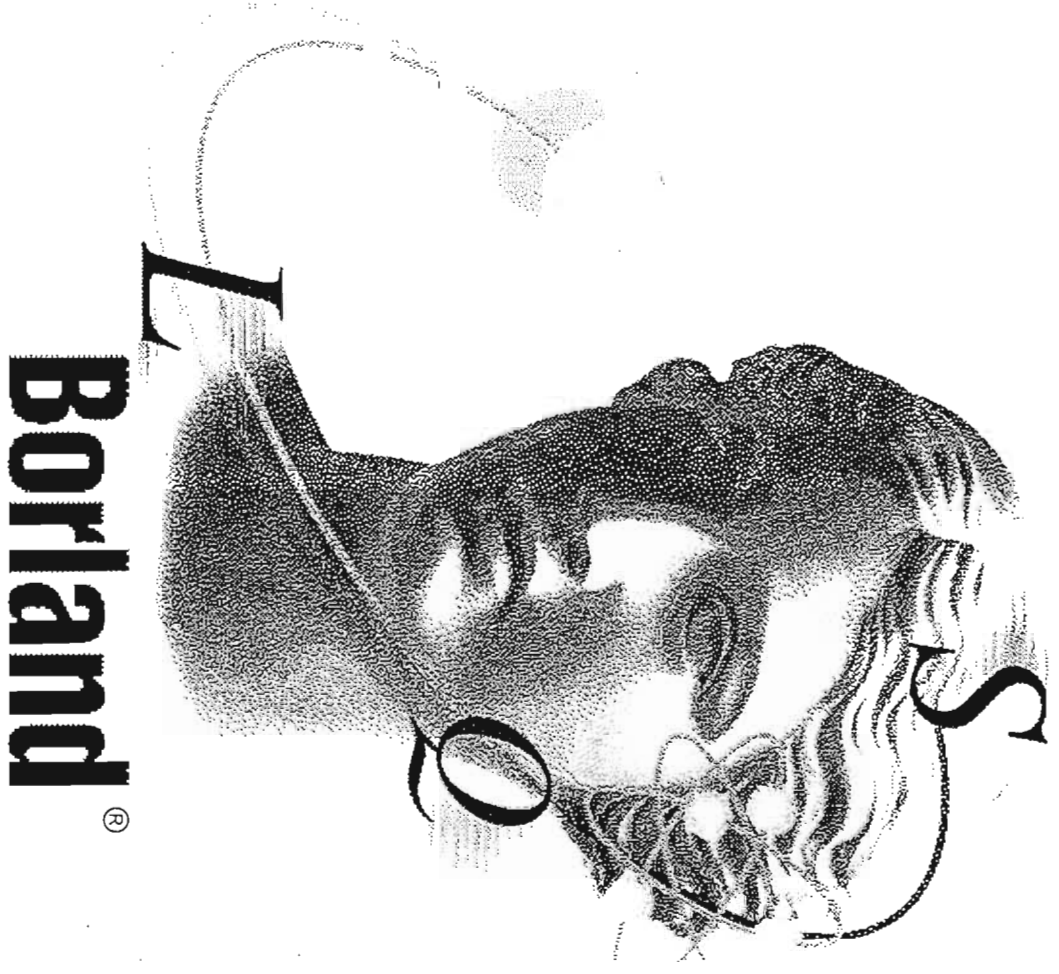
Delphi™ pour Windows



Guide du concepteur de composants

Delphi™

1E0R1294
9495969798-987654321
W1

Guide du concepteur de composants

Delphi pour Windows

Introduction

Delphi n'est pas qu'un simple environnement de développement pour créer visuellement des applications à partir de composants. Il comprend aussi tout ce qu'il faut pour créer des composants qui serviront à construire des applications dans ce même environnement, en utilisant le même langage Pascal objets.

Le Guide du concepteur de composants de Delphi et le fichier d'aide associé (CWG.HLP) décrivent tout ce qu'il faut savoir pour écrire des composants destinés aux applications Delphi. Le manuel imprimé et le fichier d'aide contiennent les mêmes informations, mais présentées différemment.

Cet ouvrage a deux objectifs :

1. Apprendre à créer des composants opérationnels
2. Apprendre à créer des composants ayant un comportement qui s'inscrit dans l'environnement Delphi

Que vous écriviez des composants pour vos propres applications ou bien en vue d'une distribution commerciale, cet ouvrage explique comment écrire des composants qui s'intègrent parfaitement dans une quelconque application Delphi.

Qu'est-ce qu'un composant ?

Les composants sont les éléments de base d'une application Delphi. Bien que la plupart des composants représentent la partie visible d'une interface utilisateur, un composant peut aussi correspondre aux éléments non visibles d'un programme tels qu'une horloge ou une base de données.

Il y a trois différentes définitions auxquelles penser concernant les composants : fonctionnelle, technique et pratique.

Définition fonctionnelle du terme "composant"

Du point de vue de l'utilisateur final, un composant est un élément qui peut être choisi dans la palette puis utilisé dans une application en la manipulant depuis le concepteur de fiche ou à partir du code. Mais, du point de vue de celui qui l'écrit, un composant est un objet dans le code. Bien qu'il y ait quelques restrictions sur ce qu'il est possible d'accomplir avec un composant, il est bon de garder en mémoire ce qu'en attend l'utilisateur final lorsqu'il utilise les composants que vous avez écrits.

Avant de commencer à écrire un composant, il vous est fortement recommandé de vous familiariser avec les composants Delphi existants pour définir correctement le "dialogue" entre vos composants et les utilisateurs. Dans toute la mesure du possible, votre objectif doit être de construire des composants qui présentent un effet "tactile" semblable à celui des composants standard.

Définition technique du terme "composant"

Au niveau le plus simple, un composant est un objet descendant du type *TComponent*. *TComponent* définit le comportement de base de tous les composants tel que leur capacité à s'afficher dans la palette des composants et à fonctionner dans le concepteur de fiche.

Mais, au-delà de cette simple définition, plusieurs points sont à prendre en considération. Par exemple, bien que le type *TComponent* définisse le comportement de base permettant leur fonctionnement dans l'environnement Delphi, il ne peut savoir comment gérer tous les ajouts spécifiques que vous serez amené à faire dans vos composants. Il faudra donc les spécifier vous-même.

Même s'il n'est pas difficile de créer des composants dont le comportement est correct, vous devez prêter tout particulièrement attention aux standards et conventions utilisés dans cet ouvrage.

Définition du terme "composant" par l'auteur de composant

A un niveau très pratique, un composant est un élément qui se "branche" dans l'environnement de développement. Il peut représenter une entité quelconque, qu'elle que soit sa complexité, allant d'une simple addition à l'interfaçage complexe d'un vaste système matériel et logiciel. En résumé, un composant correspond à ce qui peut être codé et qui obéit au cadre de développement de composants Delphi.

Définir un composant revient donc à spécifier une interface. Ce manuel décrit le cadre de développement à partir duquel vous construirez votre code spécialisé pour qu'il fonctionne dans Delphi.

Les limites d'un "composant" sont celles de la programmation. Il n'est pas plus possible d'énumérer toutes les différentes sortes de composant que vous pouvez créer, qu'il n'est possible de décrire tous les programmes que vous pouvez écrire à l'aide d'un langage déterminé. Par contre, il est possible de vous indiquer comment écrire votre code pour qu'il s'inscrive parfaitement dans l'environnement Delphi.

Qu'est-ce qui change dans l'écriture de composant ?

Il existe trois différences importantes entre la création d'un composant Delphi et la tâche plus usuelle qui consiste à créer une application utilisant des composants :

- L'écriture de composant est non visuelle
- L'écriture de composant implique une bonne connaissance des objets
- L'écriture de composant doit respecter certaines conventions

L'écriture de composant est non visuelle

La différence la plus évidente entre l'écriture de composant et la construction d'une application dans Delphi est la suivante : l'écriture de composant se résume exclusivement à écrire du code. Comme la conception visuelle des applications Delphi nécessite des composants achevés, la création de ces mêmes composants ne peut se faire qu'en écrivant du code Pascal objet.

Même si vous ne pouvez pas utiliser les outils visuels pour créer des composants, vous pouvez utiliser toutes les fonctionnalités de l'environnement de développement Delphi, y compris l'éditeur de code, le débogueur intégré et le scruteur d'objet.

L'écriture de composant implique une bonne connaissance des objets

Hormis la programmation non visuelle, la différence majeure entre la création et l'utilisation de composants est la suivante : lorsque vous créez un nouveau composant, vous devez dériver un nouvel objet à partir d'un objet existant en ajoutant des propriétés et des méthodes nouvelles. Par contre, les utilisateurs de composant utilisent des composants existants et personnalisent leur comportement au moment de la conception en changeant des propriétés et en spécifiant les réponses à apporter aux événements.

Lorsque vous dérivez de nouveaux objets, vous avez accès à des parties des objets ancêtres qui, par ailleurs, restent inaccessibles aux utilisateurs finals de ces mêmes objets. Considérées collectivement, ces parties, destinées aux auteurs de composant, représentent *l'interface protégée* des objets. Dans l'implémentation des objets descendants, les objets ancêtres sont souvent appelés. Les auteurs de composant doivent donc être familiarisés avec cet aspect de la programmation objet.

L'écriture de composant doit respecter certaines conventions

L'écriture d'un composant s'assimile à une tâche de programmation usuelle. Contrairement à la création visuelle d'applications, vous devez respecter certaines conventions. La première chose à faire avant même de commencer à écrire votre premier composant consiste à examiner les composants fournis avec Delphi pour comprendre les concepts de base tels que les conventions utilisées pour les noms ou les fonctionnalités que tout utilisateur s'attend à trouver dans vos composants.

La fonctionnalité la plus importante que tout utilisateur attend d'un composant est la suivante : à n'importe quel moment, il doit pouvoir entreprendre une quelconque action sur ce composant. Il n'est pas forcément difficile de tenir compte de cet impératif lors de l'écriture de composants, mais cela demande un certain travail de planification et le respect de certaines conventions.

Création d'un composant (résumé)

Brièvement résumé, le processus de création d'un composant se ramène à ces quelques étapes :

1. Création d'une unité pour le nouveau composant.
2. Dérivation d'un nouveau type de composant à partir d'un type existant.
3. Ajout de propriétés, de méthodes et d'événements en fonction des besoins.
4. Recensement de votre composant dans Delphi.
5. Création d'un fichier d'aide associé à votre composant et à ses propriétés, ses méthodes et ses événements.

Toutes ces étapes sont abordées en détail dans ce manuel. Votre composant terminé doit inclure les quatre fichiers suivants :

1. Une unité compilée (fichier .DCU)
2. Un bitmap de palette (fichier .DCR)
3. Un fichier d'aide (fichier .HLP)
4. Un fichier de mots-clés de l'aide (fichier .KWF)

Seul le premier fichier est obligatoire, cependant, les autres fichiers rendront vos composants à la fois plus utiles et plus exploitables.

Que contient cet ouvrage ?

Le Guide du concepteur de composants de Delphi est divisé en deux parties. La première explique tous les aspects de la construction de composants. La deuxième fournit plusieurs exemples complets montrant l'écriture de différentes sortes de composants.

Partie I, "Création de composants"

Les chapitres de cette partie de l'ouvrage décrivent les différents constituants d'un composant et montrent comment le créer.

- **Le chapitre 1, "Présentation générale de la création d'un composant"**, explique les étapes de base intervenant dans la création de tout composant. Vous devez lire ce chapitre avant de commencer à créer un composant.
- **Le chapitre 2, "Présentation objet et écriture de composants"**, présente les points de la programmation objet qu'un auteur de composant doit maîtriser.
- **Le chapitre 3, "Création de propriétés"**, présente les procédures pour ajouter des propriétés à des composants.
- **Le chapitre 4, "Création d'événements"**, décrit le processus pour ajouter des événements à des composants.
- **Le chapitre 5, "Création de méthodes"**, explique le processus pour ajouter des méthodes à des composants et décrit les conventions qu'un auteur de composant doit respecter pour nommer et définir le niveau de protection des méthodes.

- **Le chapitre 6, "Les graphiques et les composants"**, décrit les différents aspects de l'encapsulation Delphi des graphiques, particulièrement utile aux auteurs de composant.
- **Le chapitre 7, "Gestion des messages"**, décrit le système de message Windows et les mécanismes intégrés dans les composants Delphi pour gérer les messages.
- **Le chapitre 8, "Recensement de composants"**, présente les contraintes liées à la personnalisation de l'interaction de vos composants avec l'environnement de développement Delphi, y compris la façon de fournir de l'aide aux utilisateurs de composant.

Partie II, "Composants d'exemple"

Les chapitres de cette partie de l'ouvrage donnent des exemples complets montrant la construction de composants.

- **Le chapitre 9, "Modification d'un composant existant"**, montre le moyen le plus simple pour créer un composant en apportant des modifications à un composant déjà opérationnel.
- **Le chapitre 10, "Création d'un composant graphique"**, montre un exemple de création de composant en partant de rien.
- **Le chapitre 11, "Personnalisation d'une grille"**, montre comment créer un composant basé sur l'un des types abstraits de la bibliothèque de composants.
- **Le chapitre 12, "Rendre un contrôle sensible aux données"**, montre comment transformer un contrôle existant en contrôle sensible aux données.
- **Le chapitre 13, "Transformation d'une boîte de dialogue en composant"**, explique comment transformer une fiche opérationnelle en composant boîte de dialogue réutilisable.
- **Le chapitre 14, "Intégration d'une boîte de dialogue dans une DLL"**, montre comment intégrer une fiche avec ses contrôles dans une bibliothèque liée dynamiquement (DLL) afin de la rendre exploitable par une quelconque application Windows.

Aspects non traités dans cet ouvrage ?

Bien que cet ouvrage aborde tous les aspects pour définir un composant Delphi, il ne peut couvrir tous les aspects des différentes sortes de composants que vous pouvez vouloir écrire. Si vous créez un composant de bas niveau, vous devrez maîtriser les opérations de bas niveau du système.

Par exemple, pour créer des composants qui exploitent les fonctions de communication complexes intégrées dans Windows, vous devez maîtriser les communications et les fonctions de l'API Windows qui les implémentent pour effectuer les appels convenables à l'intérieur de vos composants. De même, si vous accédez aux données qui se trouvent dans une base de données non supportée par le moteur de base de données Borland, vous devez être capable de programmer l'interface de cette base de données pour que vos composants puissent en proposer l'accès.

Par contre, si votre seul objectif est de créer des versions légèrement différentes des composants standard fournis avec Delphi, il suffit simplement d'avoir une bonne connaissance de l'environnement de développement Delphi et de ses composants standard et d'avoir de solides notions de programmation.

Conventions utilisées dans le manuel

Les manuels de Delphi utilisent une typographie et des symboles spéciaux décrits dans le tableau Intro.1 pour mettre en évidence certaines parties du texte.

Tableau Intro.1 Typographie et symboles utilisés dans ce manuel

Type ou symbole	Signification
`Espacements fixes`	Un texte à espacements fixes représente ce qui apparaît à l'écran ou dans un code en Pascal objets. Il sert également à représenter ce que vous devez saisir.
[]	Les crochets dans une partie de texte ou de syntaxe indiquent des éléments facultatifs. Un texte de cette sorte ne doit pas être tapé mot à mot.
Gras	Les mots en gras dans le texte ou dans un programme représentent les mots réservés du Pascal objets ainsi que les options de compilation.
Italique	Les mots en italique dans le texte représentent les identificateurs Pascal objets, tels que les noms de variable ou de type. L'italique est également utilisé pour mettre en évidence certains mots tels que les nouveaux termes.
Touche	Ce type de caractère indique une touche sur votre clavier. Par exemple, "Frappez *Echap* pour quitter le menu."
■	Ce symbole indique le début de la description d'une procédure. Le texte qui suit décrit un ensemble d'étapes en vue d'effectuer une tâche particulière.
➤	Ce symbole indique une action spécifique à entreprendre telle qu'une étape dans un exemple.

Table des matières

Introduction 1
Qu'est-ce qu'un composant ?. 1
Définition fonctionnelle du terme "composant". 1
Définition technique du terme "composant" . .2
Définition du terme "composant" par l'auteur de composant 2
Qu'est-ce qui change dans l'écriture de composant ? 2
L'écriture de composant est non visuelle 3
L'écriture de composant implique une bonne connaissance des objets 3
L'écriture de composant doit respecter certaines conventions 3
Création d'un composant (résumé) 4
Que contient cet ouvrage ? 4
Partie I, "Création de composants" 4
Partie II, "Composants d'exemple" 5
Aspects non traités dans cet ouvrage ? 5
Conventions utilisées dans le manuel 6

Partie I
Création de composants 7

Chapitre 1
Présentation générale de la création d'un composant 9
La bibliothèque des composants visuels. 9
Composants et objets 10
Comment créer un composant. 10
Modification de contrôles existants 11
Création de contrôles originaux 12
Création de contrôles graphiques 12
Sous-classement de contrôles Windows 12
Création de composants non visuels 13
Contenu d'un composant. 13
Suppression des dépendances 14
Propriétés, événements et méthodes 15
Propriétés 15
Evénements 15
Méthodes 16
Encapsulation des graphiques 16
Recensement. 16
Création d'un nouveau composant 17
Création manuelle d'un composant 17
Utilisation de l'expert composant 19
Test des composants non installés 20
Résumé 21

Chapitre 2
Programmation objet et écriture de composants 23
Création de nouveaux objets 23
Dérivation de nouveaux types 24
Déclaration d'un nouveau type de composant. 25
Ancêtres et descendants 26
Hiérarchies d'objets 26
Contrôle d'accès. 27
Masquer les détails d'implémentation. 27
Définir l'interface de développement 28
Définir l'interface d'exécution. 29
Définir l'interface de conception 30
Méthodes de répartition. 30
Méthodes statiques 31
Méthodes virtuelles 31
Objets et pointeurs 33
Résumé 33

Chapitre 3
Création de propriétés 35
Pourquoi créer des propriétés ? 35
Types de propriétés. 36
Publication des propriétés transmises en héritage 37
Définition des propriétés de composant 38
Déclaration d'une propriété 38
Stockage interne des données. 39
Accès direct. 40
Méthodes d'accès 40
Méthode read 40
Méthode write 41
Valeurs par défaut d'une propriété 41
Spécification d'aucune valeur par défaut . . 42
Création de propriétés tableau. 42
Ecriture d'éditeurs de propriété 43
Dérivation d'un objet éditeur de propriété. . 44
Modification de la propriété sous une forme textuelle 45
Affichage de la valeur de la propriété 45
Définition de la valeur de la propriété. . . . 45
Modification globale de la propriété. 46
Spécification des attributs de l'éditeur 47
Recensement de l'éditeur de propriété 48
Résumé 49

Chapitre 4
Création d'événements 51

Qu'est-ce qu'un événement ?. 51
Les événements sont des pointeurs sur des méthodes. 52
Les événements sont des propriétés 53
Types de gestionnaire d'événement. 53
Les types de gestionnaire d'événement sont des procédures 54
Les gestionnaires d'événement sont facultatifs. 54
Implémentation des événements standard . . 55
Qu'est-ce qu'un événement standard ?. 56
Evénements standard pour tous les contrôles. 56
Evénements standard pour les contrôles standard 56
Rendre visibles des événements. 56
Modifier la gestion d'événement standard . . 57
Définition de vos propres événements. 57
Déclenchement de l'événement 58
Deux sortes d'événement 59
Définition du type de gestionnaire 59
Notifications simples 59
Gestionnaires d'événement spécifiques . . . 60
Renvoi d'informations à partir du gestionnaire 60
Déclaration de l'événement 60
Les noms d'événement débutent par "On". 60
Appel de l'événement. 60
Résumé. 62

Chapitre 5
Création de méthodes 63

Eviter les interdépendances 63
Nom des méthodes 64
Protection des méthodes 65
Les méthodes qui doivent être publiques . . . 65
Les méthodes qui doivent être protégées . . . 65
Les méthodes qui doivent être privées 66
Rendre des méthodes virtuelles. 66
Déclaration des méthodes 66
Résumé. 67

Chapitre 6
Les graphiques et les composants 69

Présentation des graphiques dans Delphi. . . 69
Utilisation du canevas. 71
Travailler avec les représentations. 71
Images, graphiques et canevas 71
Graphiques inclus dans un fichier 72
Gestion des palettes. 73
Bitmaps hors-écran 74
Création et gestion de bitmaps hors-écran . . 74
Copie d'images bitmap. 75
Réponse aux changements 76
Résumé 77

Chapitre 7
Gestion des messages 79

Compréhension du système de gestion des messages 79
Que contient un message Windows ? 80
Méthodes de répartition 80
Suivi du flux des messages 81
Modification de la gestion de message. 81
Surcharge de la méthode du gestionnaire . . 81
Utilisation des paramètres de message 82
Interception des messages 82
Création de nouveaux gestionnaires de message 83
Définition de vos propres messages 83
Déclaration d'une nouvelle méthode de gestion de message 85
Résumé 85

Chapitre 8
Recensement de composants 87

Recensement des composants dans Delphi . . 87
Déclaration de la procédure Register 88
Implémentation de la procédure Register . . 88
Ajout des bitmaps de palette. 89
Fournir l'aide des propriétés et des événements. 89
Gestion des requêtes d'aide par Delphi. . . . 90
Fusion de votre aide avec celle de Delphi . . 90
Stockage et chargement de propriétés 93
Le mécanisme de stockage et de chargement . 94
Spécification des valeurs par défaut. 94
Détermination de ce qui doit être stocké . . . 95
Initialisation après chargement 96
Résumé 96

Partie II
Composants d'exemple 97

Chapitre 9
Modification d'un composant existant 99

Création et recensement du composant . . . 100
Modification de l'objet composant 100

Surcharge du constructeur 101
Spécification de la nouvelle valeur par défaut de la propriété 102
Résumé 102

Chapitre 10
Création d'un composant graphique 103

Création et recensement du composant . . . 103
Publication des propriétés reçues en héritage 104
Ajout de fonctionnalités graphiques 105
Détermination de ce qui doit être dessiné. . 105
Déclaration du type de la propriété 105
Déclaration de la propriété 106
Ecriture de la méthode d'implémentation .106
Surcharge du constructeur et du destructeur. 106
Modification des valeurs par défaut de propriété. 106
Publication du crayon et du pinceau. 107
Déclaration des champs objet 107
Déclaration des propriétés d'accès 108
Initialisation des objets avec propriétaire . .109
Définition des propriétés des objets avec propriétaire 109
Dessin de l'image du composant 110
Adaptation du dessin de la forme 111
Résumé 112

Chapitre 11
Personnalisation d'une grille 113

Création et recensement du composant . . . 113
Publication des propriétés reçues en héritage 114
Modification des valeurs initiales 115
Redimensionnement des cellules 116
Remplissage des cellules 117
Suivi de la date. 117
Stockage de la date interne 117
Accès au jour, au mois et à l'année 118
Génération des numéros du jour 119
Sélection du jour courant. 122
Navigation de mois en mois et d'année en année 123
Navigation de jour en jour 124
Déplacement de la sélection 124
Fournir un événement OnChange 124
Exclusion de cellules vides. 125
Résumé 126

Chapitre 12
Rendre un contrôle sensible aux données 127

Création et recensement du composant . . . 128
Fonctionnement en lecture seulement du contrôle 128
Ajout de la propriété ReadOnly 128
Autorisation des mises à jour nécessaires . .129
Ajout du lien de données. 130
Déclaration du champ objet. 131
Déclaration des propriétés d'accès 131
Initialisation du lien de données 132
Réponse aux changements de données . . . 133
Résumé 134

Chapitre 13.
Transformation d'une boîte de dialogue en composant 135

Définition de l'interface du composant. . . . 135
Création et recensement du composant . . . 136
Création de l'interface du composant 137
Inclusion de l'unité de la fiche 137
Ajout des propriétés d'interface 137
Ajout de la méthode Execute 138
Vérification du fonctionnement du composant 140
Résumé 140

Chapitre 14
Intégration d'une boîte de dialogue dans une DLL 141

Ajout de la routine d'interface 142
Déclaration de la routine d'interface. 143
Implémentation de la routine d'interface. . .143
Modification du fichier projet 144
Ouverture de la boîte de dialogue à partir d'une application 145
Ouverture de la boîte de dialogue à partir d'une application Delphi. 145
Ouverture de la boîte de dialogue à partir d'une application Paradox. 146
Ouverture de la boîte de dialogue à partir d'une application dBASE 147
Ouverture de la boîte de dialogue à partir d'une application C/C++ 147
Résumé 148

Annexe A

Changements dans Pascal objets 149

Le nouveau modèle objet 149
Changements dans les déclarations d'objets . 150
Ancêtre par défaut 150
Parties protégées 150
Parties publiées 151
Méthodes de classe 151
Prédéclaration d'objet 152
Changements dans l'utilisation des objets. . 152
Modèle de référencement 152
Pointeurs de méthode 153
Références de type objet 154
Informations de type accessibles à l'exécution 155
Propriétés 155
Syntaxe de propriété 156
Champs pour lire et écrire 157
Propriétés tableau 157
Propriétés tableau par défaut 158
Propriétés à index multiples 159
Propriétés accessibles en lecture ou écriture seulement 159
Changements dans la répartition de méthode 159
Méthodes dynamiques 159
Méthodes de gestion de message 160
Gestionnaire de message par défaut 161
Méthodes abstraites 161
Directive Override 162
Constructeurs virtuels 162