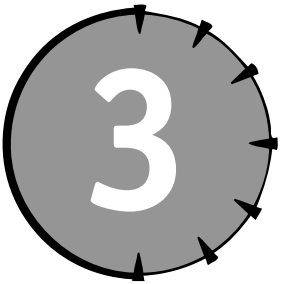
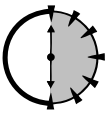
SESSION



Création de l'interface utilisateur

Programme de la session

- ✔ Les propriétés du composant TextBox
- ✔ Création d'une série de boutons par clonage
- ✔ Ajout de bulles d'aide
- ✔ Exécution et test de l'application
- ✔ Sauvegarde des fichiers de l'application
- ✔ Raccourcis clavier et propriété TabIndex



30 Min.

A partir de cette session, nous allons réaliser pas à pas une application de traitement de texte léger. La progression détaillée vous assure un bénéfice pédagogique certain. En maîtrisant l'écriture du texte source (les instructions), vous saurez personnaliser votre version en changeant la police d'édition, en augmentant la taille de la fenêtre de saisie et en effectuant d'autres aménagements : VB est toujours prêt à répondre à vos désirs.

Nous allons commencer par concevoir les grandes lignes de l'interface en plaçant les composants visuels. Dans la session suivante, nous écrirons nos premières lignes pour donner vie à l'interface (quelques lignes suffisent en VB pour créer un petit traitement de texte complet).

Préparation d'un composant TextBox

Démarrez si nécessaire l'environnement VB .NET et choisissez la commande *Fichier* ⇨ *Nouveau* ⇨ *Projet* (ou cliquez sur *Nouveau projet*) une fois dans la fenêtre de démarrage *Start* de Visual Studio. Dans le champ *Nom* de la boîte *Nouveau projet*, saisissez MotsDoux puis double-cliquez sur l'icône de la catégorie *Application Windows* (il s'agit de la catégorie des applications Windows normales ; elle démarre par un formulaire Form1 vide). Cliquez sur *OK* pour valider vos choix et fermer la boîte.

Pour que la fenêtre d'application soit plus vaste, fermez les fenêtres *Sortie* et *Résultats* ainsi que toute autre fenêtre éventuelle du coin inférieur gauche de l'environnement. Etirez le formulaire en largeur et en hauteur pour qu'il aille presque recouvrir l'Explorateur de solutions et la fenêtre *Propriétés* (Figure 3-1).

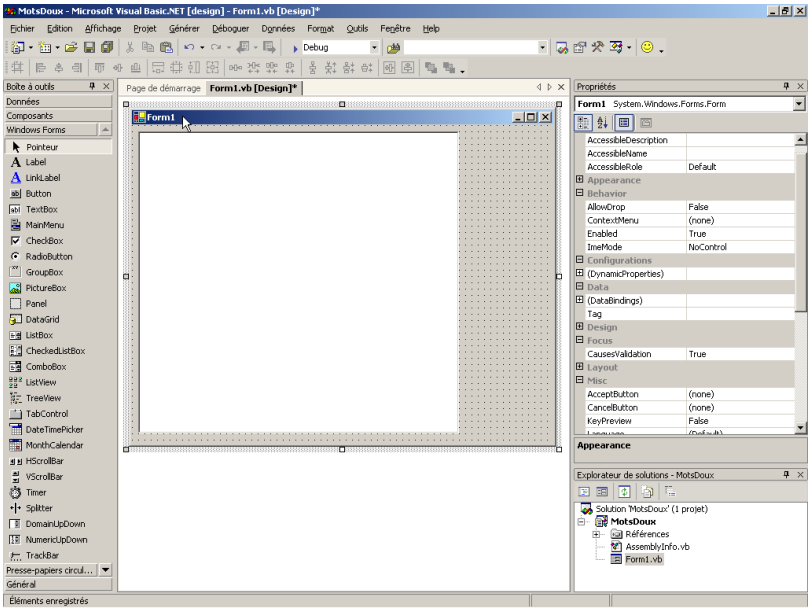


Figure 3-1
Agrandissez le formulaire pour donner du confort à la saisie.

Le premier composant requis dans un traitement de texte est évidemment une zone de saisie de texte. Dans VB .NET, cette fonction est représentée par le composant TextBox. Dans la Boîte à outils, double-cliquez sur l'icône TextBox et accédez à sa propriété MultiLine pour changer la valeur en True (de manière à pouvoir agrandir le composant). Agrandissez ensuite le composant pour qu'il occupe bien la partie gauche du formulaire (Figure 3-1).

Il faut presque toujours intervenir sur trois propriétés lorsqu'on met en place une zone de texte : Text, Font et MultiLine. En standard (par défaut), TextBox contient au départ un petit texte de légende (qui indique le nom technique initialement attribué à ce composant, TextBox1). Nous lui préférons une zone vierge. Cliquez dans le composant pour le sélectionner puis dans la ligne de sa propriété Text dans la fenêtre de propriétés. Sélectionnez le contenu par zonage et appuyez sur la touche SUPPR pour tout effacer. Le texte disparaît immédiatement du composant sur le formulaire. Il en va ainsi pour presque toutes les modifications de propriétés : elles sont répercutées aussitôt dans le formulaire.

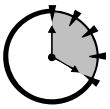
Passons à la police de caractères en vigueur pour la saisie. Celle proposée (MS Sans Serif 8 points) nous semble trop petite. Sélectionnez la propriété Font (dans la fenêtre de propriétés) et cliquez sur le bouton de dépliement à gauche (contenant un signe +) pour accéder aux sous-propriétés de police. Sélectionnez une police Times New Roman (la police à empattement standard sous Windows) et optez pour un corps 11 ou 12.

Un grave problème peut se présenter dans la version bêta française. Il sera peut-être corrigé dans la version finale. Lorsque vous changez le corps d'une police de composant *via* la boîte de dialogue *Police* puis sauvegardez et fermez le projet, la nouvelle valeur est mal stockée dans le fichier source .vb **du formulaire correspondant (*form1.vb* au départ).**
Si vous avez sélectionné le corps 11 *via* la boîte standard, Visual Studio stocke la valeur sous la forme 11.25!**, ce qui l'empêche de pouvoir relire le fichier source lorsque vous voulez réouvrir votre projet ou votre solution, et ce à cause du séparateur point au lieu de la virgule. Très désagréable !**
Si cela vous arrive, chargez le fichier source .vb **dans un éditeur autonome et remplacez chaque occurrence de la valeur** xx.25! **(dans** New System.Drawing.Font**) par une valeur entière comme** 11!**, ou bien remplacez le point séparateur par une virgule. Le projet pourra à nouveau être ouvert.**

Pour éviter que le problème apparaisse, ouvrez les détails de la sous-catégorie de propriétés *Police* (*Appearance* ⇨ *Font*) pour corriger éventuellement la valeur de corps dans sa propre ligne Size**.**

Passons à la propriété MultiLine vue plus haut. Si la valeur est False, la zone de texte n'accepte d'afficher la saisie que sur une ligne. Si vous poursuivez la saisie après avoir atteint le bord droit, le contenu existant défile vers la gauche. Il n'y a pas de retour à la ligne, même en faisant ENTREE. Cette ligne unique accepte jusqu'à 10 000 mots environ (environ 60 000 signes en français ou une trentaine de pages comme celles de notre ouvrage).

Le composant TextBox **de VB .NET est limité à la saisie de textes courts. Pour un roman fleuve ou vos mémoires, optez pour le composant** RichTextBox**, limité seulement par l'espace mémoire disponible ! Ce composant n'est pas accessible directement dans la Boîte à outils. Vous devez ouvrir les extensions grâce au bouton flèche, en bas de la Boîte à outils, puis faire défiler pour visionner les autres composants de la page *Formulaire Windows* (*Windows Form*). Nous ne l'utiliserons pas dans cet ouvrage qui se veut didactique, car** TextBox **est plus simple à mettre en place.**



20 Min.

Une fois notre composant de saisie paré, nous allons ajouter des boutons action (comme les boutons *OK* et *Annuler*, quasiment universels dans les boîtes de dialogue) pour permettre à l'utilisateur de déclencher différentes commandes en rapport avec cette saisie.

Nous allons adopter une approche différente (mais complémentaire) de celle utilisée dans les applications classiques : au lieu de faire émettre les commandes par choix dans des menus déroulants, nous allons les déclencher suite au clic de tel ou tel bouton. En théorie, une action liée à un bouton est en général de petite envergure puisqu'elle dépend de ce contexte particulier. Les commandes plus globales sont davantage à leur place dans la barre de menus (c'est que nous verrons à la Session 10).

Notre projet de traitement de texte est une sorte d'outil léger, qui se passera de barre de menus. Pour vos projets, vous choisirez de répartir les accès aux commandes entre barre de menus et boutons action selon vos préférences. L'expérience vous aidera. Si le nombre de commandes est important, vous n'aurez pas le choix, les boutons occupant de l'es-

pace dans les fenêtres : une barre de menus peut réunir une centaine de commandes sur une seule ligne !

Ajout des autres boutons

A la Session 2, nous avons découvert la technique de mise en place de composants par clonage à partir d'un premier composant dont les propriétés avaient d'abord été adaptées.

Pour notre exemple, nous désirons créer toute une série verticale de boutons pour les différentes commandes (*Enregistrer*, *Ouvrir*, *Importer*, *Exporter*, *Quitter*, etc.). Nous allons commencer par bien définir notre premier bouton, duquel seront dérivés les autres.

1. Double-cliquez sur l'icône *Button* dans la Boîte à outils pour insérer un nouveau bouton dans le formulaire.
2. Agrippez le bouton et amenez-le vers le coin inférieur droit de la fenêtre de conception (celle contenant le formulaire) sans qu'il touche les bords et ne recouvre plus l'objet TextBox.
3. Elargissez légèrement le bouton.
4. Augmentez le corps de sa police (propriété Font), choisissez la police Arial et saisissez un corps de 11 points dans Size.

Ce premier bouton sera l'incontournable bouton *Quitter* qui sert à sortir de l'application. Voyons les autres boutons requis. Puisqu'il s'agit de manipuler du texte, il faut pouvoir sauvegarder ce qui a été saisi sur texte et le recharger : il nous faut un couple de boutons *Ouvrir* et *Enregistrer* et un bouton *Nouveau*. A des fins pédagogiques, prévoyons un bouton pour importer le contenu du Presse-papiers (coller) et un autre pour copier vers le Presse-papiers. Bien sûr, les raccourcis CTRL+C ou CTRL+V offrent déjà ces deux fonctions. En guise d'exercice, nous allons ajouter un bouton pour provoquer le démarrage de l'accessoire Bloc-notes (Notepad). N'oublions pas un bouton pour imprimer le texte et un dernier bouton pour accéder aux options pour pouvoir changer la police et le corps.

Récapitulons. Il nous faut neuf boutons : *Quitter*, *Nouveau*, *Ouvrir*, *Enregistrer*, *Copier*, *Coller*, *Bloc-notes*, *Imprimer* et *Options*.

Pour plus de confort, nous allons changer les noms d'origine des composants. Commencez par changer le nom du seul bouton présent qui se nomme actuellement Button1. Rebaptisez-le btnQuitter (sélectionnez le

bouton, cliquez sur la ligne de sa propriété Name dans la fenêtre de propriétés et saisissez le nouveau nom). Profitez-en pour personnaliser la légende du bouton : cliquez sur la propriété Text et remplacez Button1 par Quitter.

Notre bouton générateur est prêt. Nous pouvons procéder à la création des huit autres boutons par clonage :

1. Cliquez sur le bouton pour le sélectionner.
2. Utilisez la combinaison CTRL+C pour le copier.
3. Utilisez la combinaison CTRL+V pour insérer un clone du bouton.
4. Le clone apparaît par-dessus l'original car il a hérité de toutes ses propriétés. Seule celle de position a été suffisamment augmentée pour qu'on puisse apercevoir l'original dessous.
5. Déplacez le nouveau bouton pour l'amener tout en haut du formulaire, dans le même alignement vertical que le premier (Figure 3-2).
6. Changez la propriété Name du nouveau bouton pour qu'elle indique btnNouveau.
7. Changez la propriété Text du nouveau bouton pour qu'elle indique Nouveau.

Il ne reste plus qu'à répéter les cinq premières étapes de ce processus pour les sept autres boutons : *Ouvrir*, *Enregistrer*, *Copier*, *Coller*, *Bloc-notes*, *Imprimer* et *Options* (avec le préfixe btn pour le nom du composant).

Procédez dans un ordre logique et en respectant les notions de blocs fonctionnels. Par convention, les commandes *Nouveau*, *Ouvrir* et *Enregistrer* se présentent dans cet ordre et sont regroupées. En revanche, la commande de sortie *Quitter* sera placée tout en bas de la colonne (elle s'y trouve déjà). Les deux boutons liés au Presse-papiers seront regroupés et séparés des autres au milieu du bandeau vertical. Le bouton d'accès au Bloc-notes sera à part. Les deux derniers boutons *Imprimer* et *Options* pourront se placer juste avant le bouton *Quitter* (Figure 3-2).

Une fois les boutons en place, accédez tour à tour à chacun d'eux pour personnaliser les deux propriétés Text et Name. Pour le nom, utilisez le préfixe btn suivi du nom de la commande qui sert de légende (il faut bannir les lettres accentuées dans les noms des composants, mais, ici, un heureux hasard fait que le problème ne se pose pas). Les huit noms de composants seront donc dans l'ordre : btnNouveau, btnOuvrir,

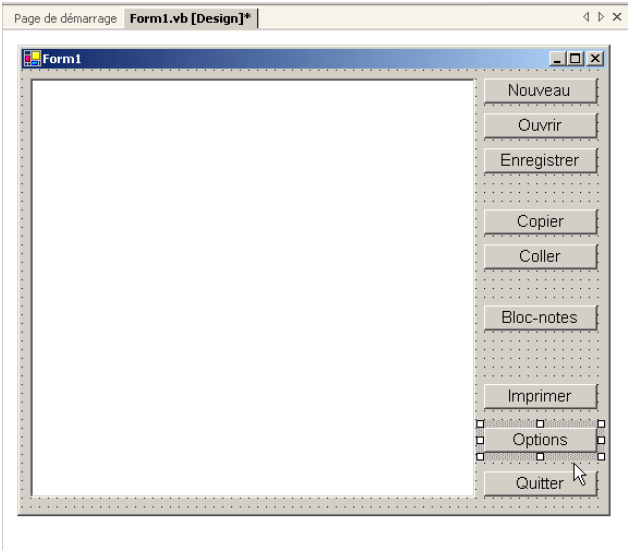


Figure 3-2
Aspect initial de l'interface de notre application de traitement de texte.

btnEnregistrer, btnCopier, btnColler, btnBlocnotes, btnImprimer et btnOptions.

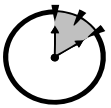
Pensez enfin à forcer l'alignement horizontal : sélectionnez tous les boutons par zonage (faites glisser le pointeur avec le bouton de souris enfoncé autour des boutons et relâchez), puis choisissez la commande *Format* ⇨ *Aligner* ⇨ *Côté gauche*.

Pour peaufiner la position d'un composant, sélectionnez-le puis maintenez la touche CTRL enfoncée tout en utilisant la touche fléchée du curseur correspondant à la direction de micro-déplacement désirée. Cette technique fonctionne également avec plusieurs composants sélectionnés.

Une fois les composants bien positionnés, mieux vaut les verrouiller. En effet, vous n'êtes jamais à l'abri d'un geste malencontreux avec la souris qui pourrait détruire votre bel agencement. Choisissez pour cela la commande *Verrouiller les composants* dans le menu *Format* (ou par un clic droit dans le formulaire). Vérifiez que le verrouillage est activé en sélectionnant un composant : vous ne devez plus voir apparaître les

huit carrés de manipulation, ce qui veut dire que vous ne pouvez plus déplacer le composant, ni le retailler.

Ajout de bulles d'aide (infobulles)



10 Min.

Les *bulles d'aide* (ou *infobulles*) sont très appréciées des utilisateurs pour les guider dans une application, notamment en phase d'apprentissage. Ce sont ces courtes descriptions qui apparaissent en surimpression d'un composant lorsque vous laissez le pointeur de souris quelques instants dans ses limites sans cliquer. Les bulles d'aide constituent un aide-mémoire très pratique.

Pour ajouter des infobulles aux composants d'interface de votre application, vous travaillez en deux étapes. Vous devez d'abord ajouter au formulaire un composant non visuel nommé ToolTip. Faites ensuite dérouler le contenu de la liste des composants de la catégorie *Formulaires Windows* (*Windows Form*) jusqu'à repérer ce composant et double-cliquez pour l'associer au formulaire actif. A partir de là, tous les composants concernés disposent d'une propriété supplémentaire nommée ToolTip. Accédez-y et saisissez un bref descriptif pour chaque bouton.

Certains composants comme ToolTip**, ainsi que la minuterie** Timer **que nous découvrirons plus tard, ne sont pas visibles dans l'interface. Ils sont d'ailleurs dits *non visuels*. Seul le programmeur les visualise dans une fenêtre spéciale, sous la fenêtre de conception. Notez aussi que la mise en place des bulles d'aide a changé par rapport à VB6.**

Test de l'application

Voyons maintenant si notre ébauche d'application VB fonctionne déjà. En lançant son exécution, nous allons pouvoir juger exactement de l'aspect qu'elle offrira à ses utilisateurs.

Pour lancer l'exécution (précédée de la compilation), appuyez sur la touche F5. Au bout de quelques instants, vous voyez apparaître la fenêtre de l'application. Cliquez sur les boutons : ils s'enfoncent et remontent comme pour une véritable application Windows. Ce sont de vrais

boutons action Windows générés à partir de ceux posés dans le formulaire (Figure 3-3). Bien sûr, les clics n'ont aucun effet pour l'instant.

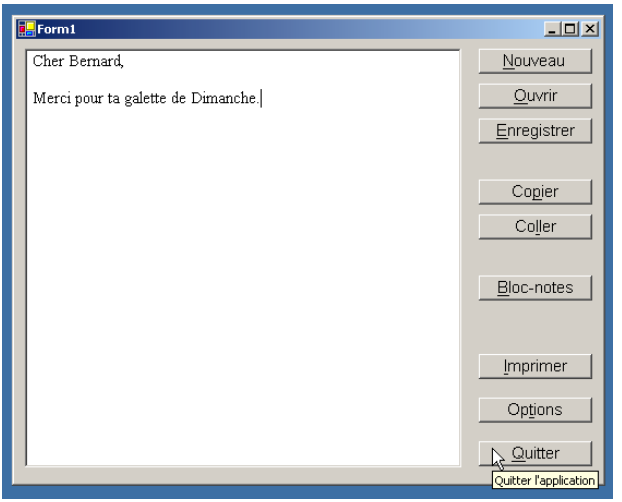


Figure 3-3
L'application fonctionne déjà : les boutons peuvent être cliqués, même s'ils ne sont pas encore raccordés à du code de traitement.

Vous avez remarqué, en bas, l'affichage automatique de la fenêtre intitulée *Sortie* dès que vous lancez l'exécution. Elle montre l'avancement de la compilation et présente les erreurs éventuellement détectées. (Nous reverrons cette fenêtre à la Session 20.)

Dès que vous avez fini d'explorer le superbe résultat de votre premier programme, utilisez la case de fermeture du coin supérieur droit de la fenêtre (l'icône *X*) ou choisissez dans l'environnement la commande *Déboguer ⇨ Arrêter le débogage*.

Sauvegarde des fichiers du projet

Après nous être consacrés à la réalisation de notre projet, voyons comment stocker toutes les définitions de composants dans des fichiers sur disque pour parer à tout problème éventuel (foudre, panne de disque

dur, blocage système). Nous pourrons ainsi reprendre en toute quiétude à la Session 4 sans repartir du début.

Pour procéder à la sauvegarde du projet, choisissez la commande *Fichier ⇨ Enregistrer tout*. Sachez que, sauf mention contraire, (en utilisant le bouton *Parcourir* de la boîte d'enregistrement), les fichiers sont placés dans le dossier Documents and Settings\ (nomutilisateur)\ Mes documents\ Projets Visual Studio.

(Nous verrons plus tard que les projets Web sont stockés dans un sous-répertoire de *Inetpub*.)

Aménagements clavier de productivité

Pour prétendre avoir fait un tour assez complet des modalités de création d'une interface utilisateur, il nous reste à voir les possibilités d'interaction *via* le clavier, souvent plus efficaces qu'à la souris, une fois la phase de mémorisation franchie. Le pourcentage d'utilisateurs privilégiant le clavier à la souris n'est pas énorme, mais ce sont les utilisateurs les plus importants, car ils cherchent à acquérir une véritable intimité avec votre programme. C'est encore plus vrai dans le cadre d'une application telle que celle de l'exemple, qui propose de saisir des données au clavier. Deux mécanismes sont disponibles : la circulation parmi les composants *via* la touche TAB et l'attribution d'une "lettre active" appelée *accélérateur* à chacune des commandes.

Ordre de circulation via la propriété TabIndex

Rappelons qu'il est possible de passer d'un composant à un autre dans une fenêtre ou une boîte *via* la touche TAB. Chaque frappe de cette touche fait changer de composant ciblé par les actions clavier (le composant ayant le "focus"). L'ordre de circulation parmi les composants est défini par la propriété numérique TabIndex que possèdent tous les composants sélectionnables. Le fait d'appuyer sur la barre ESPACE équivaut à cliquer dans le composant ayant le focus. S'il s'agit d'un bouton action, vous le voyez réellement s'enfoncer et se relever comme lors d'un clic.

Normalement, lorsque vous insérez un composant en partant de la Boîte à outils, son numéro d'ordre TabIndex augmente de 1 automatiquement par rapport au dernier inséré. Mais souvenez-vous que nous avons mis en place huit boutons en tant que clones d'un premier bouton. Dans ce cas, les valeurs ne sont pas modifiées dans TabIndex. Il nous faut donc

intervenir dans chaque bouton. L'ordre choisi est l'ordre visuel descendant. Le composant de saisie TextBox avait reçu une valeur 0 pour TabIndex. Nous la laissons telle quelle (les valeurs commencent à 0, non à 1). Cela signifie que pendant l'exécution, c'est la zone de saisie qui sera présélectionnée au départ. C'est exactement ce que nous désirons, puisque, dans ce cas, le curseur de saisie clignotant sera visible dans la zone, prêt à la saisie de texte. (La saisie dans le composant n'est d'ailleurs possible que lorsque c'est le composant sélectionné, ce que la plupart des utilisateurs font en cliquant dans la zone avec la souris.)

Nous allons mettre de l'ordre dans les propriétés TabIndex de nos boutons. Sélectionnez le premier bouton du haut (*Nouveau*) et cliquez dans la ligne de sa propriété TabIndex. Vous constatez qu'elle indique 1 (valeur héritée par clonage du bouton *Quitter*) ; nous la laissons telle quelle. Accédez ensuite à la propriété TabIndex des autres boutons et donnez une valeur ascendante entre 2 (pour *Ouvrir*)et 9 (pour *Quitter*).

Si vous voulez qu'un composant particulier ne fasse pas partie de la guirlande de sélections par TAB, forcez sa propriété TabStop **à la valeur** False**. Notez enfin que vous pouvez circuler en sens inverse *via* la combinaison de touches MAJ+TAB.**

Accélérateurs via le signe &

Les accélérateurs sont des combinaisons de touches basées sur la touche ALT et la touche de la lettre soulignée dans le nom d'un bouton, d'une commande ou d'un composant. Par exemple, le menu *Fichier* de toutes les applications sous Windows voit sa lettre F soulignée. Cela signifie que vous pouvez utiliser la combinaison ALT+F pour ouvrir le menu, ce qui équivaut exactement au clic dans le nom de menu. L'existence d'accélérateurs est très appréciée des utilisateurs chevronnés qui évitent de passer de la souris au clavier en permanence, cause de stress et d'erreurs. Ne confondez pas ces accélérateurs avec les raccourcis clavier, qui leur ressemblent soit, mais qui se basent sur des touches conventionnelles (par exemple, ALT+F4 provoque la fermeture d'une application).

Pour définir un accélérateur, il suffit d'ajouter le signe "&" devant la lettre du texte du composant qui doit devenir la lettre active. La seule contrainte est d'éviter d'attribuer la même lettre active à deux composants (ce n'est pas une cause d'erreur d'exécution, mais ça donne un air peu professionnel au programme).

Modifiez la propriété Text de chacun des neuf boutons comme ceci : &Nouveau, &Ouvrir, &Enregistrer, Co&ller, Co&pier, &Bloc-notes, &Imprimer, Op&tions et &Quitter. (Nous avons choisi l pour *Coller* et p pour *Copier* car ce sont les accélérateurs standardisés du Presse-papiers ; de même, nous avons choisi t pour *Options*, O et p étant déjà attribués.)

Procédons, comme cela doit devenir une habitude, à une sauvegarde de l'ensemble de la solution grâce à la commande *Fichier* ⇨ *Enregistrer tout*.

Lancez l'exécution *via* la touche F5. Essayez de circuler parmi les composants *via* la touche TAB pour vérifier que vous avez bien ordonné les valeurs TabIndex. Testez un accélérateur : utilisez la combinaison ALT+F4 pour quitter le programme et revenir à l'environnement. Tout doit fonctionner comme prévu.

Terminé !

RÉVISION

Nous avons fait ici une première approche pratique des modalités de création de programmes dans Visual Basic .NET. Vous avez pu voir avec quelle facilité vous pouviez mettre en place une zone d'édition de texte. Vous savez ainsi quels sont les avantages du développement rapide RAD, vu comment intervenir sur les propriétés des composants, comment compiler et exécuter le projet et comment ajouter des fonctions de productivité telles que les infobulles et les accélérateurs clavier.

Dans la prochaine session, nous allons écrire du code source pour donner vie à l'application, en associant des traitements aux boutons conformément à leurs noms.

TESTEZ VOS CONNAISSANCES

1. Citez trois caractéristiques de Visual Basic .NET qui participent à l'approche de développement rapide RAD. (Voir le début de la session.)
2. Citez les trois propriétés qu'il faut presque toujours modifier lorsqu'on met en place un composant de saisie TextBox. (Voir "Préparation d'un composant TextBox".)
3. Quel est l'intérêt du mécanisme de clonage de composant ? (Voir "Ajout des autres boutons".)

4. A quoi servent les accélérateurs clavier appelés *lettres actives* ? (Voir "Aménagements clavier de productivité".)

5. Quel symbole devez-vous ajouter à la propriété Text d'un composant pour définir un accélérateur clavier ? (Voir "Aménagements clavier de productivité".)