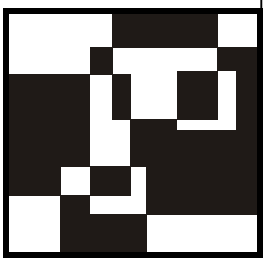
ГЛАВА 13



Программирование в Outlook

13.1. Зачем программировать в Outlook

Outlook (вместе с его урезанной версией, которая называется Outlook Express) — самая распространенная программа для работы с электронной почтой. Однако его важность заключается не только в возможности отправки и получения электронных сообщений. На предприятиях очень большую ценность представляют его дополнительные возможности, которые помогают делать то, что Microsoft называет задачами персонального информационного менеджера (Personal Information Manager, PIM).

Первая такая задача — это *работа с календарем*, т. е. организация времени пользователя. **Календарь** тесно интегрирован с другими элементами Outlook (такими как **Контакт** и **Задача**), а также с внешними приложениями (например, Microsoft Project). На предприятиях часто используется назначение задач пользователям, когда такие задачи автоматически появляются как элементы **Календаря**. Если пользователь самостоятельно заносит свои задачи в Outlook, то можно (если почтовый ящик пользователя лежит на сервере Exchange Server) предоставить доступ к календарю этого пользователя его менеджерам. Менеджеры смогут видеть, какие задания есть в настоящий момент у этого пользователя, что позволит избежать ситуаций, когда новое задание выдается сотруднику, уже занятому важной работой.

У **Календаря** есть еще одна замечательная возможность: если вы создадите из Outlook на сервере Exchange общую папку с элементами типа **Календарь**, в вашем распоряжении будет готовое приложение для планирования совместного доступа к ресурсам (комнатам для переговоров, проекторам, разному оборудованию и т. п.). Предположим, например, что в вашей организации принято правило: все личное общение с клиентами должны производиться только в комнате для переговоров, а количество этих комнат ограничено.

Вполне может произойти ситуация, когда клиент придет, а все комнаты будут заняты. Чтобы этого не произошло, используется общая папка с элементами управления типа **Календарь**. В процессе общения с клиентом пользователь открывает из Outlook эту общую папку и смотрит, в какое время комната свободна. Согласовав с клиентом время визита, он создает в этой общей папке элемент типа **Встреча**, и уже другие сотрудники смогут увидеть, что в это время комната для переговоров занята.

Конечно же, на практике как синхронизация календаря с другими источниками, так и использование общих календарей часто требует добавления функциональности средствами программирования.

Вторая задача — это *работа с контактами*. Контакты Outlook — это наиболее распространенный и стандартный формат адресной книги. В последнее время особенно важным стало то, что контакты из адресной книги Outlook можно очень легко и просто синхронизировать с сотовыми телефонами, карманными компьютерами и прочими аналогичными устройствами. На предприятиях часто используется, например, общий список контактов для подразделения в виде элементов **Контакт**, которые хранятся в общей папке Exchange Server (адресная или телефонная книга самой организации также обычно ведется в формате адресной книги Exchange Server и доступна через Outlook). Ситуаций, когда вам нужно программным образом создать контакты (например, на основе информации из базы данных), или синхронизировать их, или пройти циклом по всем контактами и изменить их в зависимости от какого-то условия, очень много.

Третья возможность — это *работа с задачами и поручениями*. Для масштабных проектов, конечно, лучше использовать специализированное программное обеспечение (например, Microsoft Project и Project Server), но для простых проектов, за которые ответственен один менеджер, задачи Outlook вполне подойдут. При помощи этого средства можно создавать задачи, назначать их другим лицам (поручения) с уведомлением их по электронной почте, отслеживать процент выполнения и т. п.

Четвертая задача — это *работа с дневником*. При помощи этого средства вы можете отслеживать работу с документами Office, обмен информацией при помощи Outlook и т. п. Дневник позволяет создать отчет о проделанной работе или вспомнить, в каком файле находится нужный документ за прошлый квартал, чтобы по его образцу сделать новый документ.

Есть еще *работа с заметками* — компьютерными аналогами маленьких листочков с напоминаниями, которыми некоторые любители обклеивают все вокруг себя.

Конечно же, перечисленными встроенными возможностями работа с Outlook не ограничивается. На связке Outlook—Exchange Server основана целая об-

ласть программирования, которая называется collaboration development — *разработка приложений коллективного использования*. Основные задачи, которые решаются при помощи приложений коллективного использования, — это сбор и автоматизированная обработка внутрикорпоративной информации.

Например, представим себе следующую задачу из реальной жизни: каждый банк должен в конце каждого месяца представлять в Центральный банк информацию об экономических нормативах. Чаще всего это выглядит так: сотрудник планово-экономического отдела, ответственный за сбор информации, в начале каждого месяца идет в бухгалтерию, чтобы получить информацию об остатках на требуемых счетах на конец прошлого месяца. Затем он отправляется в кредитный отдел, чтобы получить информацию о том, какие кредиты относятся к какой категории. После этого он руками формирует файл отчета требуемого формата. При использовании средств Outlook это могло бы выглядеть по-другому:

1. В начале месяца у сотрудника планово-экономического отдела в папке **Входящие** в Outlook автоматически появляется специальная форма для заполнения информации о нормативах.
2. Он нажимает кнопку **Отправить в бухгалтерию** на этой форме, и она отправляется ответственному сотруднику бухгалтерии (он может и автоматически получать информацию об остатках на счетах из программы **Операционный день**, если разработчик сможет это реализовать).
3. Сотрудник бухгалтерии, заполнив нужные поля, нажимает кнопку **Дальше** и форма с сохраненными данными идет в кредитный отдел.
4. Сотрудник кредитного отдела заполняет свои поля (поля, заполненные в бухгалтерии, при этом автоматически должны быть доступны только для чтения) и нажимает кнопку **Дальше**.
5. Форма со всеми необходимыми данными приходит к сотруднику планово-экономического отдела, тот нажимает на ней кнопку **Сформировать отчет**, и формируется файл отчета в нужном формате, а форма с сохраненными значениями автоматически помещается в архив.

На любом предприятии задач по сбору внутрикорпоративной информации очень много. Можно привести в пример и отчеты о командировках, сбор информации из филиалов, формы, заполняемые продавцами (в офисе и в командировках), информацию инвентаризаций — этот перечень можно продолжать бесконечно. И, как показывает практика, самый удобный способ — это именно применение средств Outlook и Exchange Server. Большим плюсом здесь является то, что никаких внешних средств разработки использовать не нужно: Outlook — это еще и среда разработки. Чаще всего в подобных приложе-

ниях используются формы Outlook — специальные шаблоны сообщений с элементами управления и программными возможностями, например, маршрутами прохождения (эти формы ни имеют никакого отношения к обычным формам VBA или формам Access). Создать такую форму и определить для нее необходимый программный код можно средствами самого Outlook: для этого достаточно в меню **Сервис | Формы** воспользоваться пунктом **Конструктор форм**, выбрать нужную форму (например, **Сообщение** в **Библиотеке стандартных форм**) и нажать кнопку **Открыть**. Откроется окно дизайнера форм (рис. 13.1), в котором вы сможете изменять шаблон стандартного сообщения, как вам угодно: помещать новые элементы управления, привязывать к ним программный код (при помощи меню **Форма | Просмотреть код**) и т. п.

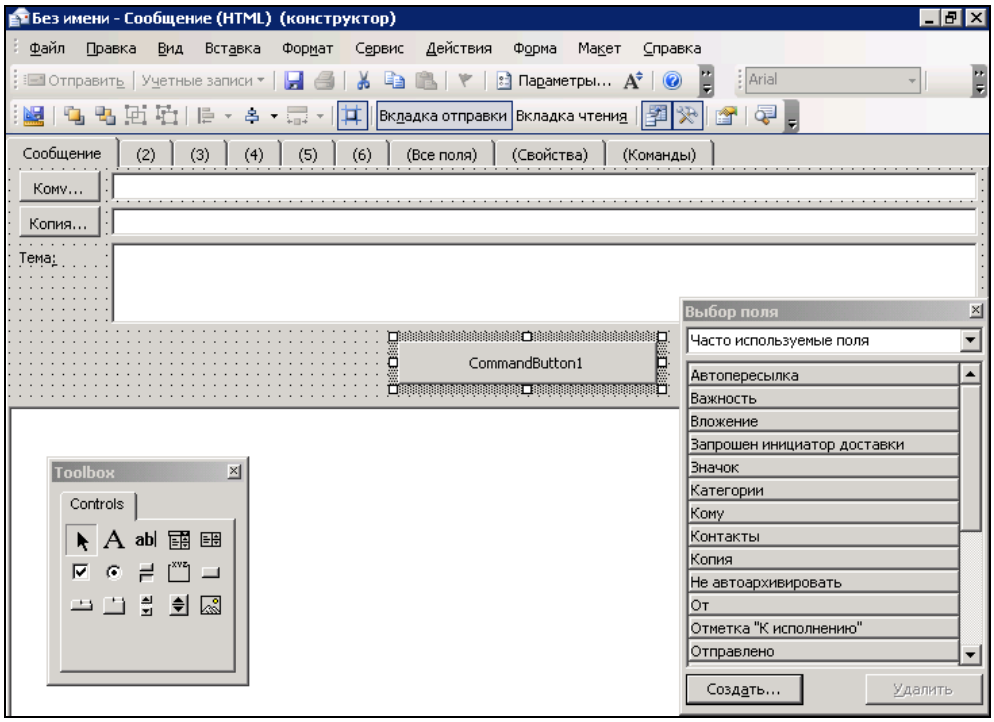


Рис. 13.1. Окно конструктора форм Outlook

Например, добавить код для нашей кнопки `CommandButton1`, которую мы поместили на форму, можно так:

1. В меню **Форма** выберите **Просмотреть код** (вместо этого можно воспользоваться кнопкой **Просмотреть код** на панели инструментов **Конструктор форм**);

2. В окне редактора сценариев добавьте следующий код:

```
Function CommandButton1_Click()
    MsgBox "Привет из формы Outlook"
End Function
```

   К сожалению, автоматически сгенерировать событийную процедуру для кнопки не получится. Нет в нашем распоряжении и подсказок по свойствам и методам, и подсветки синтаксиса, и отладки, и многого другого.

3. Чтобы запустить кнопку на выполнение, воспользуйтесь командой **Выполнить форму** в меню **Форма**, а затем нажмите кнопку.

Как вы уже, наверное, догадались, это не совсем привычный нам VBA (и даже больше — это вообще другой язык программирования VBScript). Для форм Outlook предусмотрена своя собственная среда программирования, своя объектная и событийная модели. Как правило, работа с формами Outlook неотделима от работы с корпоративными возможностями Exchange Server: библиотеками форм, серверными скриптами, общими папками, маршрутизацией и т. п. Все это очень большая специальная тема, для рассмотрения которой потребуется отдельная толстая книга. По этой причине работу с формами Outlook и приложениями коллективного использования здесь мы рассматривать не будем. Скажем только, что хорошей отправной точкой для самостоятельного освоения этой темы может послужить файл официальной документации Microsoft, который по умолчанию находится в каталоге C:\Program Files\Microsoft Office\OFFICE11\1049\OLFM10.CHM, и сайт **www.slipstick.com**. В этой главе мы сосредоточимся на работе в Outlook традиционными средствами VBA, при помощи стандартных модулей и форм VBA и в привычном редакторе кода.

Задач автоматизации, которые могут решаться средствами VBA, также очень много:

- организация рассылки электронной почты по расписанию;
- получение электронной почты и ее автоматизированная обработка (например, если сообщение пришло в стандартном формате, можно извлечь из него данные и поместить в базу данных), например, сбор данных из филиалов;
- автоматизация работы с контактами, например, импорт контактов из базы данных или файла Excel в общую папку на сервере Exchange;
- автоматическое создание или изменение записей в календаре;
- и многое-многое другое.

Работать с этими возможностями Outlook нам и предстоит научиться.

13.2. Некоторые особенности программирования в Outlook

Программирование в Outlook имеет ряд интересных особенностей, о которых необходимо упомянуть.

Первая особенность заключается в том, где именно хранятся программные модули Outlook, в которых мы создаем код. Как мы помним, в Word они хранятся вместе с документами (или шаблонами, например, Normal.dot), в Excel — в файлах рабочих книг, в Access — в файлах баз данных MDB. В Outlook информация стандартных модулей хранится в файле личных папок PST, который по умолчанию создается в профиле данного пользователя. В результате, с одной стороны, работа с программным кодом VBA в Outlook упрощается: для данного пользователя на этом компьютере он становится доступен из Outlook всегда. С другой стороны, становится труднее предоставить этот код в распоряжение другого пользователя. В этой ситуации можно использовать два выхода:

- первый выход — *воспользоваться средствами экспорта и импорта программных модулей*, которые доступны из контекстного меню для модуля в **Project Explorer** (рис. 13.2);

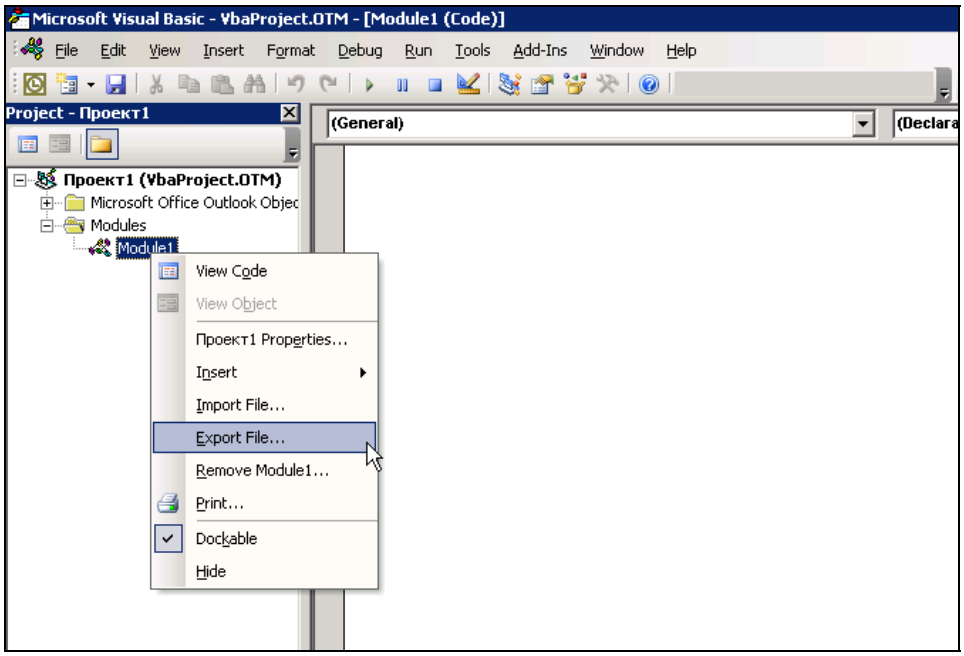


Рис. 13.2. Средства импорта и экспорта программного кода

- второй выход — *создать контейнерное приложение*, например, файл Word или книгу Outlook, из которого программным образом запускать Outlook и выполнять в нем необходимые действия.

Вторая особенность работы с VBA в Outlook заключается в том, что в Outlook реализована концепция пространства имен. Пространство имен в Outlook формально определяется как абстрактный корневой объект для любого источника данных (например, папки в почтовом ящике Exchange или PST-файле на диске). Проще всего представить себе пространство имен Outlook как некий драйвер, который нужно использовать для подключения к данным. В настоящее время Outlook поддерживает только одно пространство имен — MAPI (Messaging Application Programming Interface, интерфейс прикладного программирования для работы с сообщениями), но разработчики Outlook требуют, чтобы это пространство имен явно указывалось при выполнении самых разных операций. Подробнее про объект `Namespace`, представляющий пространство имен, будет рассказано в *разд. 13.4*. Например, для запуска Outlook и открытия в нем папки **Inbox** (Входящие) из другой программы придется использовать следующий код (не забудьте подключить ссылку на библиотеку Microsoft Outlook 11.0 Object Library):

```
Dim oOutlook As New Outlook.Application
Set oNameSpace = oOutlook.GetNamespace("MAPI")
Set oInbox = oNameSpace.GetDefaultFolder(olFolderInbox)
oInbox.Display
```

Третья особенность работы с Outlook заключается в некоторой терминологической путанице. Как правило, в документации по объектным моделям приложений Office термин `Item` (элемент) применяется к элементам коллекций. В Outlook он получает второе значение: `Item` — это все, что может храниться в папках Outlook: почтовые сообщения (объект `MailItem`), контакты (объект `ContactItem`), встречи (объект `Appointment`) и т. п. Не путайте!

У Outlook есть еще одна особенность. На протяжении многих лет Outlook была программой, которая первой подвергалась атакам вирусов, троянских программ и прочего вредоносного программного обеспечения, приходящего по электронной почте. Иногда такие атаки бывали успешными, и Outlook на компьютере пользователя сам начинал рассылать электронные письма с вирусами (в том числе и при помощи кода VBA). Чтобы снизить вероятность развития событий по такому сценарию, разработчики Outlook сознательно поместили в его объектную модель ограничения, которые должны препятствовать распространению вирусов. Иногда эти ограничения (у них есть специальное название — *Outlook Object Model Guard*) мешают и нормальной работе программ VBA. О них будет рассказано в следующих разделах. Иногда по причине таких ограничений бывает удобнее использовать вместо объектной

модели Outlook библиотеку CDO, которая имеется на любом компьютере с Windows 2000, XP или 2003.

Интересной особенностью Outlook является и то, что в отличие от других приложений Office вы не сможете напрямую (при помощи ключевого слова `New` или команды `CreateObject()`) создать ни одного объекта Outlook, кроме `Application`. Для создания всех остальных объектов придется использовать соответствующие методы уже созданных объектов.

На всякий случай также заметим, что макрорекордера в Outlook, как и в Access, к сожалению, нет. Всю необходимую дополнительную информацию вам придется искать при помощи документации.

13.3. Объект *Application*, его свойства и методы

Как и у всех приложений Office, на вершине объектной модели Outlook находится объект `Application`. Его можно использовать для запуска Outlook из внешних приложений (см. пример в предыдущем разделе). Отличительной особенностью объекта `Application` в Outlook является относительно небольшое количество свойств и методов (за счет того, что часть из них переехала в объект `Namespace`). Наиболее часто используемые свойства объекта `Application` представлены далее.

- `Explorers` — это свойство возвращает коллекцию `Explorers` с объектами `Explorer`, каждый из которых представляет собой папку Outlook, открытую на просмотр пользователем. Основное назначение этой коллекции и объектов `Explorer` — проверка, не открыта ли уже пользователем в Outlook та или иная папка, и, в зависимости от результата, активизация этого окна (`Explorer.Activate()`) или его закрытие (`Explorer.Close()`). Метод `ActiveExplorer()` объекта `Application` позволяет получить ссылку на окно, активное в настоящий момент, а `GetExplorer()` — получить ссылку на объект `Explorer` для указанной папки Outlook (без его автоматической активизации).
- `Inspectors` — свойство, которое очень похоже на `Explorers`. Оно возвращает коллекцию `Inspectors` с объектами `Inspector`. Главное отличие в том, что объекты `Inspector` представляют не открытые папки Outlook, как `Explorer`, а открытые на просмотр и редактирование элементы (например, почтовые сообщения). Объект `Inspector` используется для таких же проверок, что и объект `Explorer`, его свойства и методы почти полностью совпадают со свойствами и методами объекта `Explorer`. Для тех же целей предусмотрены и методы `ActiveInspector()` и `GetInspector()` объекта `Application`.

- `Reminders` — позволяет вернуть коллекцию `Reminders` с объектами `Reminder`, представляющими текущие оповещения. Обычно это свойство используется для того, чтобы программным образом отключить все оповещения:

```
Dim oOutlook As New Outlook.Application
Dim oReminder As Outlook.Reminder
For Each oReminder In oOutlook.Reminders
    oReminder.Dismiss
Next
```

- `Session` — это свойство позволяет вернуть объект `Namespace`, представляющий пространство имен для текущего сеанса (т. е. пространство имен MAPI). Это свойство можно использовать вместо метода `GetNamespace()`. Подробнее про объект `Namespace` будет рассказано в *разд. 13.4*.

  Аналогичное свойство `Session` предусмотрено для самого объекта `Namespace` и для множества других объектов Outlook.

Теперь расскажем о методах объекта `Outlook.Application`.

- Методы с префиксом `Active...` — просто возвращают ссылку на активный в настоящее время объект `Explorer` или `Inspector`.
- `AdvancedSearch()` — очень важный метод. Он позволяет производить поиск по папкам Outlook (что на практике требуется достаточно часто). Подробнее про этот метод и сопутствующие ему объекты `Search` и `Results` будет рассказано в *разд. 13.7*.
- `CopyFile()` — позволяет скопировать файл с диска в папку Outlook. Можно использовать, например, для переноса всех файлов из каталога с документацией по проекту в общую папку Exchange Server или в библиотеку документов SharePoint Portal Server.
- `CreateItem()` — метод, который используется очень часто. Он позволяет создать новые элементы в Outlook. Например, создать новый элемент типа контакт, заполнить его свойства, сохранить, а затем открыть для просмотра можно так:

```
Dim oOutlook As New Outlook.Application
Dim oContact As Outlook.ContactItem
Set oContact = oOutlook.CreateItem(olContactItem)
oContact.FirstName = "Академия специальных курсов"
oContact.Email1Address = "info@askit.ru"
oContact.Save
oContact.Display
```

А теперь представьте, что вы создаете объекты контактов в цикле на основе записей из базы данных или строк в таблице Excel. Справочник контактов будет загружен в Outlook очень быстро и эффективно. Только не забывайте после каждого создания и сохранения контакта удалять его объект из оперативной памяти — иначе память на компьютере кончится и это приведет к ошибке. В нашем примере удалить объект контакта из памяти можно при помощи строки:

```
Set oContact = Nothing
```

- `CreateItemFromTemplate()` — точно так же создает новый элемент Outlook, но уже на основе шаблона Outlook в файловой системе — файла oft.
- `GetNameSpace()` — метод, который используется, наверное, в большинстве программ VBA в Outlook. Позволяет получить объект пространства имен MAPI. Подробнее про работу с этим объектом будет рассказано в следующем разделе.
- `IsSearchSynchronous()` — используется для проверки режима поиска *(см. разд. 13.7)*.
- `Quit()` — осуществляет выход из Outlook.

13.4. Объект *Namespace*

Как уже говорилось ранее, в Outlook используется понятие пространств имен — нечто вроде драйверов, которые, по замыслу разработчиков Outlook, должны обеспечивать доступ к различным хранилищам пользовательских данных. У каждого такого драйвера должны быть свои возможности. Однако уже на протяжении долгого времени в Outlook используется единственное пространство имен — MAPI, и пока нет никакой информации о том, что должно появиться что-то еще. Для наших целей пространство имен Outlook можно рассматривать просто как специальный служебный объект, в который "переехали" некоторые свойства и методы объекта `Application`.

Если в программе вам нужны свойства или методы объекта `Namespace`, получить ссылку на этот объект можно двумя способами:

1. Воспользоваться методом `GetNameSpace()` объекта `Application`:

   ```
   Set oNameSpace = Application.GetNamespace("MAPI")
   ```

2. Воспользоваться свойством `Session` того же объекта `Application`:

   ```
   Set oNameSpace = Application.Session
   ```

Эти две строки кода по функциональности полностью равнозначны, но в документации обычно всегда используется только первый способ.

Объект `Namespace` нужен для выполнения самых распространенных операций с электронной почтой: установка соединения с сервером электронной почты, отправка и получение электронной почты, выбор нужной папки, работа с адресными книгами и многих других. Кроме того, объект `Namespace` представляет еще и виртуальный корень папок Exchange, при помощи которого можно циклом проходить по всем папкам в Exchange.

В практической работе объект `Namespace` используется очень часто. Предположим, что вам нужно из Outlook скачать почтовые сообщения для всех учетных записей электронной почты и что-то сделать с каждым полученным сообщением. В нашем примере мы будем просто выводить тему каждого сообщения. На практике можно "разбирать" каждое сообщение в стандартном формате и помещать из него информацию в базу данных. Такое решение может пригодиться при обработке информации, получаемой из филиалов, с Web-сайта предприятия, который находится у провайдера, от торговых представителей, которые находятся в командировке, и т. п.

Как может выглядеть подобное сообщение?

Представим себе, что мы работаем из внешнего приложения. Удобнее всего нам будет использовать форму Word или Excel, чтобы работа с событиями производилась из окна редактора кода. Первое, что нужно сделать, — это поместить в проект ссылку на объектную библиотеку Microsoft Outlook 11.0 Object Library.

Затем мы занимаемся привычным делом — создаем самую обычную форму VBA и помещаем в нее кнопку `CommandButton1`. Перед созданием кода для этой кнопки необходимо сделать еще одно дело: зарегистрировать события для некоего абстрактного объекта `Outlook.Items` в разделе **General—Declarations** редактора кода. Это можно сделать при помощи строки:

```
Public WithEvents oItems As Outlook.Items
```

После этого объект `oItems` с событиями появится в окне редактора кода.

Затем можно создавать код для события `Click()` нашей кнопки. Он может быть, например, таким:

```
Private Sub CommandButton1_Click()

    'Запускаем Outlook
    Dim oOutlook As New Outlook.Application

    'Очень важно: указываем, что события oItems — это события
    'папки Inbox в Outlook
    Set oItems = _
      oOutlook.GetNamespace("MAPI").GetDefaultFolder(olFolderInbox).Items
```

```
    'Дальше просто запускаем прием почты со всех учетных записей
    Dim oNamespace As Outlook.NameSpace
    Set oNamespace = oOutlook.GetNamespace("MAPI")
    oNamespace.SyncObjects("Все учетные записи").Start

End Sub

'Ловим события появления нового сообщения
Private Sub oItems_ItemAdd(ByVal Item As Object)
    MsgBox Item.Subject
End Sub
```

В документации Microsoft для события `ItemAdd` указано, что оно может не срабатывать, если в папку одновременно добавляется большое количество элементов, но у меня он всегда срабатывал правильно.

Конечно, если у вас на предприятии установлен Exchange Server, то для обработки всей входящей электронной почты правильнее использовать серверные скрипты Exchange и его событийную модель. Но реальная жизнь, как всегда, сложнее. Например, часто за Exchange Server отвечает администратор сети, который совершенно не собирается разрешать разработчикам настраивать на нем какие-то скрипты. Другой случай — когда на вашем предприятии работает не Exchange, а, к примеру, почтовая система на UNIX — SendMail, PostFix, CommunigatePro и т. п. Чтобы разобраться с их событиями, может потребоваться много времени и усилий, а Outlook всегда под рукой.

А теперь, как обычно, рассмотрим информацию о самых важных свойствах и методах объекта `Namespace`.

- `AddressLists` — это свойство возвращает коллекцию `AddressLists`, в которой находятся объекты `AddressList`, представляющие все адресные книги, доступные в настоящий момент. Например, получить список всех доступных в данный момент для пользователя адресных книг можно так (подразумевается, что вы работаете с Outlook из внешнего приложения):

  ```
  Dim oOutlook As New Outlook.Application
  Dim oNameSpace As Outlook.NameSpace
  Dim oAddress As Outlook.AddressList
  Set oNameSpace = oOutlook.GetNamespace("MAPI")
  For Each oAddress In oNameSpace.AddressLists
      Debug.Print oAddress.Name
  Next
  ```

  В объекте `AddressList` находится коллекция `AddressEntries` с объектами `AddressEntry`, представляющими записи в адресных книгах. При помощи этой объектной "ветви" вы можете программным способом добавлять записи в адресную книгу, удалять их, изменять свойства и т. п.

Если Outlook у вас работает сам по себе, в работе с коллекцией `AddressLists` нет никакого смысла — единственной доступной для пользователя адресной книгой будет являться книга **Контакты**, с которой проще работать другим способом (при помощи объектов `ContactItem`). Но если Outlook подключен к Exchange Server, то эта возможность становится очень интересной.

- `CurrentUser` — еще одно очень полезное свойство. Возвращает информацию о текущем пользователе (от имени которого открыт Outlook) в виде объекта `Recipient`, при помощи которого можно получить, например, адрес электронной почты данного пользователя, его имя и прочие атрибуты, которые предусмотрены для записи в списке адресов (если они были определены для пользователя). Например, получить доступ к информации об адресе электронной почты текущего пользователя можно так:

```
Set oNameSpace = Application.GetNamespace("MAPI")
Set oRecipient = oNameSpace.CurrentUser
Debug.Print oRecipient.Address
```

  К сожалению, попытка выполнить этот код из внешнего приложения приведет к появлению окна сообщения с вопросом: хотите ли вы предоставить программе доступ к адресам электронной почты в Outlook? Скорее всего, это окно вам совершенно не нужно, но оно было сделано специально в целях безопасности и избежать его вам не удастся. В вашем распоряжении два варианта: программно имитировать нажатие клавиш <Shift>+<Tab> и <Enter> в этом окне (заботливые разработчики отключили даже горячую клавишу для кнопки **Да**) или запускать этот код из уже работающего Outlook, тогда предупреждения не возникнет. Для программной имитации нажатий клавиш можно использовать объект `WshShell` объектной библиотеки Windows Script Host, но поскольку при выполнении кода VBA ошибки не происходит, а просто "подвисает" последняя строка `Debug.Print`, то возникают дополнительные сложности. Их можно обойти только средствами Windows API.

- `ExchangeConnectionMode` — очень важное для практической работы свойство. Оно позволяет определить, настроен ли Outlook для работы с Exchange Server и подключен ли он к Exchange Server в настоящий момент.

- `Folders` — это, наверное, самое главное свойство объекта `Namespace`. Возвращает коллекцию `Folders` с объектами `MAPIFolder`, представляющими папки верхнего уровня в Outlook (у каждой папки, в свою очередь, есть свое свойство `Folders`, так что вы вполне можете пройти циклом по всем без исключения папкам Outlook). Кроме того, что у папок Outlook есть множество своих собственных важных свойств и методов, через свойство `Items` папки вы можете получить доступ ко всем элементам папки (сооб-

щениям, контактам, элементам календаря и т. п.). Подробнее про коллекцию `Folders` и объект `MAPIFolder` будет рассказано в следующем разделе.

- `Offline` — позволяет выяснить, подключен ли в настоящее время Outlook к серверу электронной почты или нет.
- `SyncObjects` — возвращает одноименную коллекцию с объектами `SyncObject` (объекты синхронизации). Сами эти объекты представляют собой группы отправки (то, что при помощи графического интерфейса Outlook можно найти в меню **Сервис | Отправить/Получить**). Самое важное, что можно сделать при помощи объектов `SyncObject`, — это программно инициировать соединение с сервером электронной почты или разорвать его.

Обычно в приложениях, использующих объектную модель Outlook, без методов объекта `Namespace` также не обойтись.

- `AddStore()` и `AddStoreEx()` — позволяют программно открыть файл PST хранилища сообщений Outlook на диске. Обратите внимание на то, что если такого файла на диске нет, то при вызове этого метода Outlook просто создаст его. `AddStoreEx()` отличается тем, что позволяет указать кодировку для файла сообщений. Закрыть PST-файл можно при помощи метода `RemoveStore()`.
- `CreateRecipient()` — позволяет программным образом создать объект `Recipient`. Обычно он используется для передачи в виде параметра при вызове метода `GetSharedDefaultFolder()`. Этот метод необходим для подключения к папке в чужом почтовом ящике.
- `Dial()` — позволяет открыть диалоговое окно **Новый звонок**, чтобы пользователь мог установить коммутируемое соединение. В качестве необязательного параметра принимает имя контакта, который будет автоматически подставлен в это окно.
- `GetDefaultFolder()` — важнейший метод, возвращающий объект `MAPIFolder` для одной из двенадцати встроенных (используемых по умолчанию) папок Outlook: **Входящие**, **Контакты**, **Календарь**, **Отправленные** и т. п. Например, подключиться к папке **Контакты** (и для наглядности открыть ее) можно так:

```
Dim oOutlook As New Outlook.Application
Set oNameSpace = oOutlook.GetNamespace("MAPI")
Set oInbox = oNameSpace.GetDefaultFolder(olFolderContacts)
oInbox.Display
```

- Методы `Get...FromID()` — обычно используются только тогда, когда вы переводите свой код, написанный с использованием объектной библиоте-

ки CDO (о ней — в *разд. 13.8*), на использование объектной модели Outlook.

- `GetSharedDefaultFolder()` — этот метод делает то же, что и `GetDefaultFolder()`, но применяется тогда, когда у пользователя в Outlook кроме своего ящика открыты еще и почтовые ящики других пользователей. Метод позволяет получить ссылку, например, на папку **Inbox** в другом почтовом ящике.
- `Logon()` — позволяет установить соединение по протоколу MAPI (т. е. установить соединение с Exchange Server). В качестве необязательных параметров принимает имя почтового профиля, параметр, определяющий, показывать ли пользователю диалоговое окно выбора профиля и т. п. Разрыв соединения производится при помощи метода `Logoff()`.
- `PickFolder()` — открывает диалоговое окно **Выбор папки**. Если пользователь выбрал папку в этом окне и нажал **OK**, то возвращается объект `MAPIFolder` для выбранной папки.

13.5. Коллекция *Folders* и объект *MAPIFolder*

Обычно, когда мы программным образом работаем с Outlook, нам нужно что-то сделать с его элементами — почтовыми сообщениями, контактами, встречами в календаре и т. п. Все эти элементы расположены в папках Outlook (либо встроенных, либо созданных пользователем). Папкам в объектной модели Outlook соответствуют объекты `MAPIFolder`, которые сведены в коллекцию `Folders`.

В Outlook папки могут быть вложены друг в друга. На самом верху расположены папки верхнего уровня (это не **Входящие**, **Контакты**, **Черновики** и т. п., как вы могли подумать, а папки более высокого уровня, например, **Личные папки**, **Общие папки**, **Mailbox — Administrator**). Доступ к коллекции `Folders`, представляющей собой папки самого верхнего уровня, производится через свойство `Folders` объекта `Namespace` *(см. разд. 13.4)*:

```
Dim oOutlook As New Outlook.Application
Dim oNameSpace As Outlook.NameSpace
Dim oFolder As Outlook.MAPIFolder
Set oNameSpace = oOutlook.GetNamespace("MAPI")
For Each oFolder In oNameSpace.Folders
    Debug.Print oFolder.Name
Next
```

Если вам нужна конкретная встроенная папка, например **Inbox**, то проще всего найти ее при помощи метода `GetDefaultFolder()` объекта `Namespace`:

```
Dim oOutlook As New Outlook.Application
Dim oNameSpace As Outlook.NameSpace
Dim oFolder As Outlook.MAPIFolder
Set oNameSpace = oOutlook.GetNamespace("MAPI")
Set oFolder = oNameSpace.GetDefaultFolder(olFolderInbox)
Debug.Print oFolder.Name
```

Очень часто требуется пройти циклом по всем папкам в почтовом ящике Outlook (а иногда даже в нескольких ящиках), просмотреть все сообщения в каждой папке (и во всех вложенных папках) и что-то сделать с этими сообщениями, например, собрать все письма от определенного отправителя, разбросанные по разным папкам, в одну специальную. В нашем примере мы просто пройдем циклом по всем папкам Outlook и выведем их имена, но, конечно, этот пример несложно переделать так, чтобы он выполнял какое-нибудь действие над ними в зависимости от условия. В нашем случае удобнее всего использовать две процедуры, одна из которых будет вызывать саму себя:

```
Public Sub StartProc1()
    Dim oOutlook As New Outlook.Application
    Dim oNamespace As Outlook.NameSpace
    Dim oChildFolder As Outlook.MAPIFolder
    Set oNamespace = oOutlook.GetNamespace("MAPI")

    'Перебираем каждую папку верхнего уровня и
    'вызываем для нее процедуру DoFolder()
    For Each oChildFolder In oNamespace.Folders
        DoFolder oChildFolder
    Next

End Sub

'Эта процедура выводит имя всех сложенных папок и
'для каждой из них опять вызывает саму себя
Public Sub DoFolder(ByVal oFolder As MAPIFolder)
    Dim oChildFolder As Outlook.MAPIFolder
    For Each oChildFolder In oFolder.Folders
        Debug.Print oChildFolder.Name
        'Для каждой вложенной папки опять вызываем процедуру DoFolder()
        DoFolder oChildFolder
    Next

End Sub
```

Далее представлена информация о свойствах и методах коллекции `Folders` и объекта `MAPIFolder`.

У коллекции `Folders` свойства и методы стандартные, как и у большинства коллекций (`Count`, `Item()`, `Add()`, `Remove()` и т. п.). Зато у объекта `MAPIFolder` важных свойств и методов очень много.

Самые важные свойства объекта `MAPIFolder` приведены в следующем списке.

- `AddressBookName` — позволяет поменять имя папки с контактами для отображения в адресной книге пользователя. Для других папок применяться не может (вернется ошибка).
- `CurrentView` — возвращает объект `View`, который определяет, как отображается данная папка для пользователя. Для объекта `MAPIFolder` это свойство доступно только для чтения.
- `DefaultItemType` — это свойство позволяет вернуть (в виде константного значения) тип элемента папки по умолчанию (почтовое сообщение, контакт и т. п.). Это свойство определяется при создании папки и после изменено быть не может. Обычно это свойство используется для исключения каких-то папок из обработки (чтобы, например, не искать почтовые сообщения в папке с контактами).
- `DefaultMessageClass` — то же самое, что и `DefaultItemType`, но информация возвращается не в виде числа, а в виде строкового значения. Выбирайте, что вам удобнее.
- `Description` — просто описание папки. При использовании графического интерфейса Outlook доступно через свойства папки.
- `EntryID` — это свойство очень удобно использовать как уникальный идентификатор сообщения, который создается автоматически при появлении сообщения в любом MAPI-совместимом хранилище (например, в файле Outlook или в почтовом ящике Exchange Server) и изменяется только при переносе в другое хранилище.
- `FolderPath` — полный путь к папке в иерархии хранилища Outlook, например, `"\\Личные папки\Входящие"`.
- `Folders` — очень важное свойство, которое возвращает коллекцию вложенных папок для данной папки. Как уже говорилось, очень часто используется для того, чтобы пройти циклом по всему дереву папок.
- `InAppFolderSyncObject` — определяет, будет ли эта папка синхронизироваться при работе специальной группы синхронизации (объекта `SyncObject`) под названием **Application Folders**. Эта группа синхронизации (она видна через меню **Сервис | Отправить/Получить**) отличается тем, что только ее можно изменять программным образом.
- `IsSharePointFolder` — позволяет определить, находится ли эта папка с контактами или элементами календаря на Windows SharePoint Services

(для обеспечения коллективной работы). Обычно используется для проверок.

- `Items` — еще одно важнейшее свойство. Обеспечивает доступ к коллекции `Items` всех элементов данной папки. Подробнее про работу с элементами папок — в *разд. 13.6*.
- `Name` — это, конечно, имя папки.
- `ShowAsOutlookAB` — определяет, показывать ли содержимое папки с элементами типа **Контакты** в окне выбора адреса при создании почтового сообщения. По умолчанию для всех папок с **Контактами** это свойство установлено в `True`. Обычно используется только тогда, когда какая-то папка с контактами используется для служебных целей.
- `ShowItemCount` — определяет, что будет показываться в строке сообщений приложения Outlook для папки: ничего, общее количество всех сообщений или общее количество только непрочитанных сообщений.
- `StoreID` — MAPI-совместимый уникальный идентификатор хранилища, в котором находится данная папка. Выглядит как очень длинная строка (516 символов). Можно использовать для распознавания, например, нескольких почтовых ящиков Exchange.
- `UnReadIemCount` — количество непрочитанных сообщений для данной папки. Доступно только для чтения.
- `Views` — возвращает коллекцию объектов `View` (режимов отображения), которые ассоциированы с данной папкой.
- `WebViewOn` — позволяет включить для папки отображение в виде HTML-страницы (которая может быть совершенно посторонней и никак не связанной с элементами Outlook и с самой этой папкой). Страница, которую нужно отобразить, задается при помощи свойства `WebViewURL`. Эту возможность можно включить и в графическом интерфейсе на вкладке **Домашняя страница** свойств папки.

То, что делают методы `CopyTo()`, `MoveTo()`, `Delete()`, `AddToFavorites()`, `Display()`, — понятно из их названий. Метод `GetExplorer()` позволяет вернуть объект `Explorer`, представляющий эту папку в Проводнике Outlook.

13.6. Коллекция *Items* и объекты элементов Outlook

Работа с элементами папок (почтовыми сообщениями, контактами, элементами календаря и т. п.) — это обычно самая важная часть программ, исполь-

зующих объектную модель Outlook. Именно с ними и приходится выполнять различные операции.

Доступ к элементам папок чаще всего производится через свойство `Items` объекта `MAPIFolder`, которое возвращает коллекцию `Items`. В этой коллекции находятся все элементы данной папки. Однако единого объекта для них не предусмотрено. Вместо этого в вашем распоряжении 16 отдельных объектов для каждого вида элементов в Outlook.

- `AppointmentItem` — то, что на графическом интерфейсе русского Outlook называется **Встречей**. Этот элемент обычно находится в папке **Календарь**.
- `ContactItem` — это контакт. Создавать новые контакты программным образом приходится очень часто.
- `DistList` — это еще один элемент, который обычно находится в папке **Контакты**. Он представляет собой список рассылки.
- `DocumentItem` — это любой файл, который помещен внутрь хранилища Outlook и не совпадает по своему формату ни с одним другим элементом Outlook. Объектом `DocumentItem` может быть, например, документ Word, книга Excel, ZIP-архив, файл Acrobat Reader PDF, исполняемый EXE-файл и т. п. — в общем, любой файл операционной системы. Однако увлекаться хранением файлов в хранилищах Outlook (файлах PST, а также в почтовых ящиках и общих папках Exchange Server), конечно, не стоит. Это замедлит доступ к обычным элементам Outlook. Такая возможность изначально была предусмотрена для того, чтобы, например, в общей папке Outlook для проекта вместе с перепиской по нему хранить также и различные, относящиеся к нему файлы.
- `JournalItem` — это запись в дневнике.
- `MailItem` — наиболее привычный многим элемент. Представляет собой сообщение электронной почты.
- `MeetingItem` — приглашение на встречу (специальный тип электронного сообщения). Обычно встречается там же, где и обычные сообщения электронной почты, например, в папке **Inbox**. Создать программным образом этот элемент невозможно — он создается только автоматически при получении соответствующего сообщения (отправить его можно при помощи объекта `AppointmentItem`, что соответствует элементу **Встреча** в **Календаре**).
- `NoteItem` — объект заметки (из папки **Заметки**). От всех других элементов отличается минимальным количеством свойств и методов.

- `PostItem` — еще одна специальная разновидность почтового сообщения. Это сообщение, которое отправлено в общую папку. От обычного объекта `MailItem` отличается тем, что:
  - встречается только в общих папках;
  - для его отправки вместо метода `Send()` используется метод `Post()`.
- `RemoteItem` — тоже очень специальная разновидность почтового сообщения. Этот объект представляет собой почтовое сообщение с минимальным количеством заполненных свойств (заполнена может быть только информация о получателе, отправителе, дате получения и размере сообщения) и текстом сообщения, в котором находятся первые 256 символов письма. Эти объекты создаются автоматически в тех ситуациях, когда Outlook подключается к почтовому ящику на сервере Exchange Server через соединение удаленного доступа (обычно коммутируемое). Если сообщение находится в файле OST (т. е. скачано с того же сервера Exchange Server по MAPI или получено по протоколу POP3/IMAP4), то этот объект никогда для него не создается.
- `ReportItem` — специальное почтовое сообщение, представляющее собой специально сгенерированное служебное письмо: обычно это сообщение о невозможности доставки *(non-delivery report)*, созданное вашим почтовым сервером, о задержке в передаче сообщения или о другой ошибке. Эти объекты также нельзя создавать программным образом, они создаются только автоматически при получении сообщения такого типа.
- `TaskItem` — это задача или поручение из папки **Задачи**.
- `TaskRequestAcceptItem`, `TaskRequestDeclineItem`, `TaskRequestItem`, `TaskRequestUpdateItem` — это специальные почтовые сообщения, которые относятся к переписке по поводу делегирования задач. Эти объекты также нельзя создавать программным образом.

Надо сказать, что в подавляющем большинстве случаев вас будут интересовать только объекты `MailItem`, `ContactItem` и иногда `DocumentItem`. На их рассмотрении мы и сосредоточимся.

И еще один очень важный момент. Как уже говорилось ранее, в объектную модель Outlook встроены специальные ограничения, которые призваны не дать вирусам использовать возможности этой объектной модели в своих интересах. Как правило, это самые важные свойства, в которых содержится информация об адресах электронной почты, имени отправителя, тексте писем и т. п. Эти ограничения встроены в следующие объекты: `AppointmentItem`, `ContactItem`, `MailItem` и `TaskItem`, т. е. во все основные объекты элементов Outlook. Ограничения наложены и на некоторые действия, которые могут выполняться с этими объектами, например, на отправку писем. Так, создание

и отправка электронного сообщения средствами Outlook выглядит очень просто:

```
Dim oOutlook As New Outlook.Application
Dim oMessage As Outlook.MailItem
'Создаем объект сообщения
Set oMessage = oOutlook.CreateItem(olMailItem)
'Кому
oMessage.To = "Administrator@nwtraders.msft"
'Тема сообщения
oMessage.Subject = "Привет из VBA"
'Текст сообщения. Использование свойства Body означает,
'что мы посылаем сообщение обычным текстом.
'Можно также послать сообщение в формате HTML или RTF
oMessage.Body = "Текст сообщения"
'Добавляем вложение
oMessage.Attachments.Add ("C:\installlog.txt")
'Отправляем сообщение
oMessage.Send
```

Однако чтобы усложнить жизнь вирусам, вместо простой отправки сообщения появляется окно, аналогичное представленному на рис. 13.3. Более того, чтобы добиться окончательной победы над вирусами, кнопка **Да** на протяжении нескольких секунд будет недоступна.

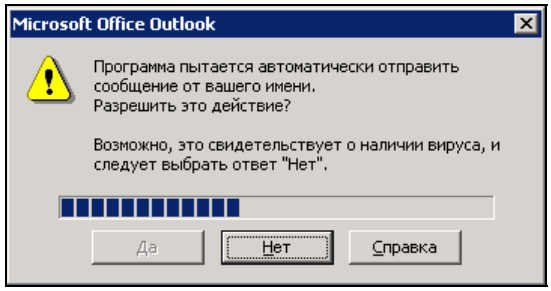


Рис. 13.3. Предупреждающее окно в Outlook

Что же делать в такой ситуации?

Варианты могут быть разными:

- первый вариант — *отправлять сообщение из макроса, который находится в модуле VBA самого Outlook* (без программного открытия Outlook). В этом случае такое окно появляться не будет. Но такое решение обычно не очень удобно, особенно с точки зрения переноса этого программного кода для запуска на разных компьютерах;

- второй вариант — *использовать специальные заменители объектов Outlook*, которые не выдают таких сообщений и позволяют разработчикам отправлять сообщения из VBA без проблем. Самое распространенное и рекомендованное средство такого рода — Outlook Redemption. Получить информацию о нем и бесплатно скачать его можно с сайта **http:// www.dimastr.com/redemption**. Этот вариант вполне удобен, если вы занимаетесь рассылкой электронной почты с одного сервера, но устанавливать его на все компьютеры, конечно, хлопотно;
- третий вариант — *программным образом обнаруживать окна подтверждений и также программно нажимать в них нужные кнопки*. Однако разработчики Microsoft позаботились о том, чтобы из кода VBA или VBScript сделать это было невозможно. В принципе, такие операции вполне можно выполнить при помощи низкоуровнего программного интерфейса Windows API (или, как вариант, получить доступ к этим возможностям при помощи специальных средств специального программного интерфейса для работы с сообщениями MAPI). Однако вам придется использоваться вместо VBA другими языками программирования (C++ или Delphi), поскольку VBA не поддерживает все нужные типы данных для работы с API. Кроме того, для уверенной работы с API требуются специальные знания в области системного программирования. Дополнительную информацию об использовании таких приемов при работе с Outlook можно получить из статьи **http://www.mapilab.com/ru/support/articles /vb_outlook_security_1.html** (на русском языке);
- четвертый вариант (с моей точки зрения наиболее удобный) — *использовать для рассылки и программной обработки входящих писем объектную библиотеку CDO*, которая есть на любом компьютере под управлением Windows 2000, XP и 2003. Подробнее о применении этой библиотеки будет рассказано в *разд. 13.8*.

Теперь о самих объектах элементов Outlook. Свойств и методов у этих объектов очень много (например, у объекта `ContactItem` почти 100 свойств), и на рассмотрение всех их потребовалось бы очень много времени. Тем не менее большинство свойств этих объектов очевидны и проблем при их использовании возникнуть не должно. Обычно единственный вопрос, который может возникнуть, — какое программное свойство соответствует определенному атрибуту элемента. К сожалению, макрорекордера в Outlook не предусмотрено, но понять, как называется то или иное свойство, можно при помощи окна **Locals**. Его использование рассмотрим на примере. Пусть вам нужно найти, какому программному свойству объекта `ContactItem` соответствует поле **Краткое имя**.

Первое, что нам нужно сделать, — это создать контакт, в котором было бы заполнено данное свойство, например, строкой `"Краткое имя"` (рис. 13.4).

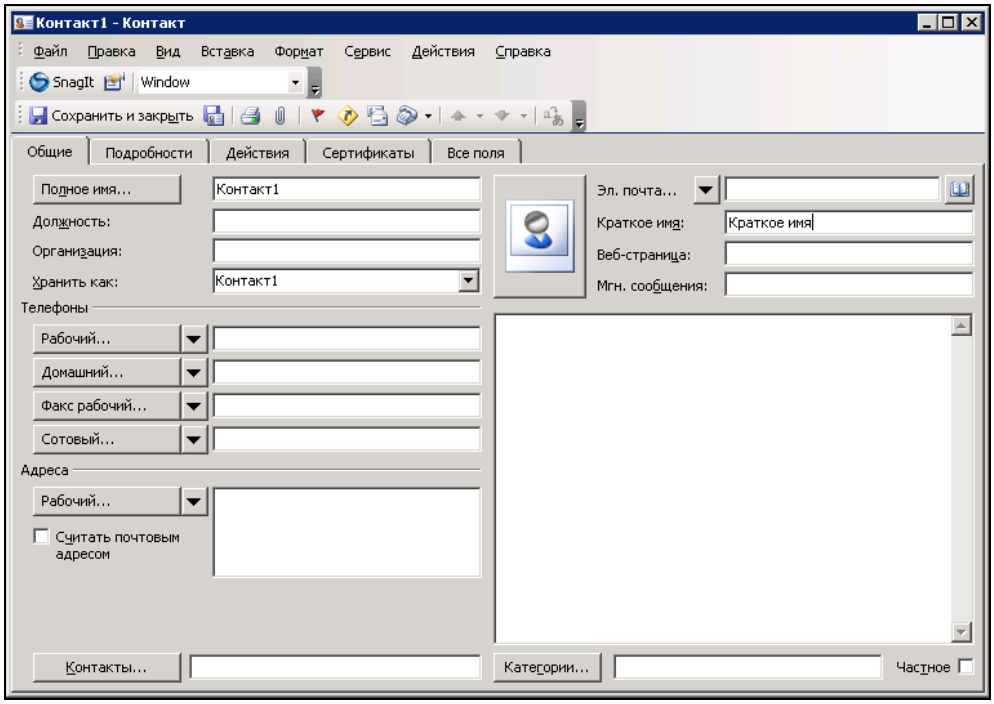


Рис. 13.4. Создаем контакт

Следующее, что нужно сделать, — написать программный код, в котором был бы создан объект данного элемента. Например, если этот контакт находится в папке **Контакты**, то код (из макроса в Outlook) может быть таким:

```
Dim oNamespace As NameSpace
Dim oFolder As MAPIFolder
Dim oContactItem As ContactItem

Set oNamespace = Application.GetNamespace("MAPI")
Set oFolder = oNamespace.GetDefaultFolder(olFolderContacts)
Set oContactItem = oFolder.Items("Контакт1")
Stop
```

Команда `Stop` здесь предназначена для того, чтобы остановиться в нужном месте.

Затем нужно запустить код на выполнение и, когда он остановится на последней строке, открыть (при помощи меню **View**) окно **Locals**.

В нашем случае вы увидите в нем четыре объекта (рис. 13.5).

Конечно же, нам нужно развернуть узел объекта `oContactItem` и поискать его свойство, для которого установлено значение `"Краткое имя"`. К своему

удивлению, мы можем обнаружить, что это свойство называется `EMail1DisplayName` (рис. 13.6).

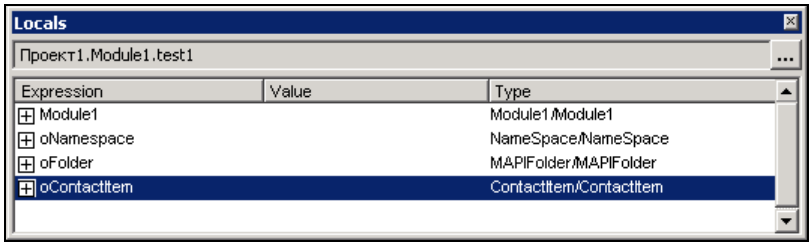


Рис. 13.5. Окно **Locals** с объектом контакта

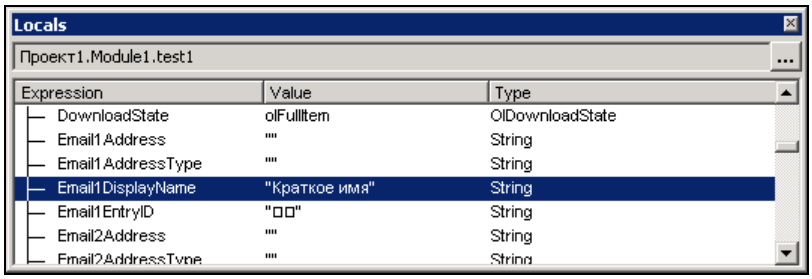


Рис. 13.6. Свойство с заполненным нами значением в окне **Locals**

Таким же образом можно находить нужные свойства и для других объектов `Item`.

13.7. Другие объекты Outlook

Мы прошли с вами только по главной ветви объектной модели Outlook: `Application` — `Namespace` — `Folders/MAPIFolder` — *`объекты_элементов`*. Но за ее пределами находится еще очень много удобных в применении объектов, о которых будет рассказано в этом разделе.

Объект `Explorer` предназначен для отображения папки в интерфейсе Outlook. При этом вы можете выбрать для отображения встроенное или пользовательское представление (при помощи объекта `View`). Например, чтобы программным образом открыть папку **Контакты** и рассортировать в ней объекты контактов по организациям, можно использовать следующий код:

```
Dim oNamespace As NameSpace
Dim oFolder As MAPIFolder
Dim oExplorer As Explorer
```

```
Set oNamespace = Application.GetNamespace("MAPI")
Set oFolder = oNamespace.GetDefaultFolder(olFolderContacts)
Set oExplorer = oFolder.GetExplorer()
oExplorer.Display
oExplorer.CurrentView = "По организациям"
```

Просмотреть все представления, доступные для данной папки (для них используется объект `View`), можно так:

```
Dim oNamespace As NameSpace
Dim oFolder As MAPIFolder
Dim oView As View

Set oNamespace = Application.GetNamespace("MAPI")
Set oFolder = oNamespace.GetDefaultFolder(olFolderContacts)
For Each oView In oFolder.Views
    Debug.Print oView.Name
Next
```

Еще одна полезная возможность объекта `Explorer` — определение того, какие элементы выделил пользователь. Например, чтобы получить информацию о темах всех писем, которые в настоящее время выделил пользователь в текущем окне, можно использовать код:

```
For Each Item In Application.ActiveExplorer.Selection
    If TypeName(Item) = "MailItem" Then Debug.Print Item.Subject
Next
```

Брат-близнец объекта `Explorer` — объект `Inspector`. Он также представляет окно Outlook, но уже с открытым на просмотр или редактирование элементом (почтовым сообщением, контактом и т. п.). Получение на него ссылки выглядит точно так же, как и для объекта `Explorer`, но используется он реже, в основном для проверок, не открыт ли обрабатываемый программно элемент пользователем в Outlook.

Отдельную ветвь представляют объекты для поиска в Outlook. Производить поиск в Outlook приходится очень часто. Можно реализовать свой собственный поиск простым перебором всех папок и элементов в них, начиная с верхнего уровня. Такой пример был приведен в *разд. 13.5*, посвященном работе с коллекцией `Folders` и объектами `MAPIFolder`. Более эффективная, но несколько более сложная возможность — это применение для поиска встроенных средств Outlook. Для этого используются два объекта — `Search` и `Results`, один метод `AdvancedSearch()` объекта `Application` и одно событие `AdvancedSearchComplete` (поскольку поиск работает в асинхронном режиме и

можно одновременно запускать до 100 поисков, как программно, так и из графического интерфейса). Выглядит это следующим образом.

Вначале запускаем на выполнение метод `AdvancedSearch()`. Этот метод принимает четыре параметра:

- `Scope` — диапазон поиска, т. е. имя папки. Если вы точно не знаете, где вам нужно искать, можно получить имя папки программно при помощи свойства `Folders` объектов `NameSpace` и `MAPIFolder`. При использовании специальных символов в имени папки рекомендуется заключать его в одинарные кавычки;
- `Filter` — самый сложный параметр. Определяет, что именно мы будем искать. Синтаксис должен соответствовать синтаксису фильтров при запросах на SQL Server (можно использовать в том числе и замечательный оператор `LIKE`), что, в принципе, могло быть очень удобным. Однако определить, как должны выглядеть имена столбцов, не так-то просто — скорее всего, за справкой придется обращаться на сайт Microsoft по адресам:
  - **http://msdn.microsoft.com/library/en-us/cdosys/html/_cdosys_schema_mailheader.asp**;
  - **http://msdn.microsoft.com/library/en-us/cdosys/html/_cdosys_schema_httpmail.asp**.

  Например, если мы производим поиск по теме сообщения (пытаясь найти все письма, в названиях которых встречается слово "Отчет"), то строка фильтра должна выглядеть так:

  ```
  "urn:schemas:mailheader:subject LIKE '%Отчет%'"
  ```

  Если же ищем точную тему "Отчет", то так:

  ```
  "urn:schemas:mailheader:subject = 'Отчет'"
  ```

- `SearchSubFolders` — значение этого параметра нужно установить в `True`, если вам нужно пройти и по всем вложенным папкам.
- `Tag` — просто строковый идентификатор поиска. Используется только тогда, когда вы одновременно запускаете несколько поисков, и вам нужно отличить результаты одного от результата другого.

В итоге метод `AdvancedSearch()` вернет нам объект `Search`:

```
Dim oSearch As Search
Set oSearch = Application.AdvancedSearch("Inbox", _
    "urn:schemas:mailheader:subject LIKE '%Отчет%'", True, "Search1")
```

Поскольку поиск выполняется в асинхронном режиме, процедура, из которой мы вызвали метод `AdvancedSearch()`, продолжит выполняться дальше, не до-

жидаясь его завершения. А нам придется отслеживать окончание поиска при помощи события `AdvancedSearchComplete`. Для этого в **Project Explorer** раскройте контейнер **Microsoft Office Outlook Objects**, щелкните два раза левой кнопкой мыши по строке **ThisOutlookSession** и в списке объектов и событий в верхней части окна редактора кода выберите `Application` и `AdvancedSearchComplete()` (рис. 13.7).

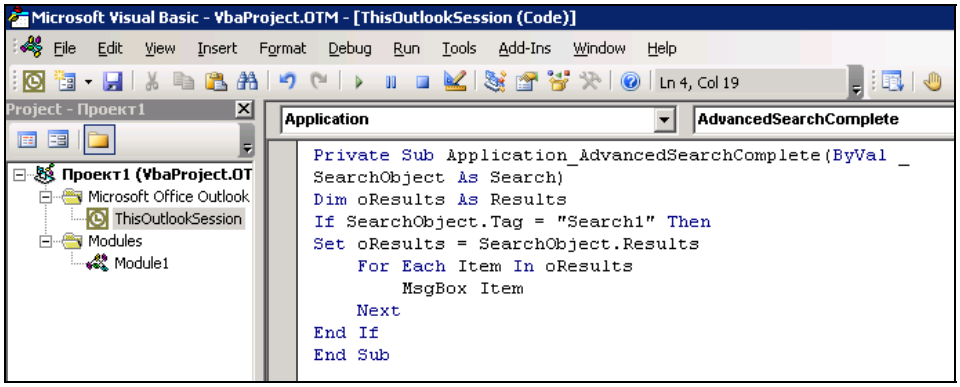


Рис. 13.7. Используем событие `AdvancedSearchComplete`

А в этом событии используем объект `Search` и производный от него объект `Results` (код показан на этом же рисунке). `Item` — это обычные элементы Outlook (в данном случае объекты `MailItem`), и вам вполне доступны все их свойства и методы.

13.8. Альтернатива при работе с электронной почтой — объектная библиотека CDO

Все-таки Outlook — это, прежде всего, программа для работы с электронной почтой, и если вы обратились к ее объектной модели, то для автоматизации именно операций с электронной почтой. Как уже было сказано ранее, нормальной работе с электронной почтой сильно мешают ограничения безопасности, встроенные в объектную модель Outlook. С ними можно бороться (как описано в *разд. 13.6*), а можно просто обойти, используя для отправки электронной почты специальную объектную модель CDO, в которой этих ограничений нет. Эту объектную модель можно использовать в том числе и из Outlook.

CDO (*Collaboration Data Objects*, объекты для совместной работы с данными) — это специальный набор библиотек для работы с электронной почтой и для администрирования сервера Exchange Server. Существует множество вер-

сий и разновидностей библиотек, которые входят в набор CDO, но нас интересует только одна: Microsoft CDO for Exchange 2000 Library, которая устанавливается вместе с Microsoft Office. Первое, что нужно будет сделать, — это добавить ссылку на эту библиотеку при помощи меню **Tools | References** в редакторе VBA.

Самый простой вариант отправки почты средствами CDO выглядит так:

```
Dim oMyMail As New CDO.Message
oMyMail.To = "Administrator@nwtraders.msft"
oMyMail.From = "Administrator@nwtraders.msft"
oMyMail.Subject = "Hello from CDO"
oMyMail.TextBody = "Our letter"
oMyMail.AddAttachment "C:\1.txt"
oMyMail.Send
```

Однако этот вариант с параметрами по умолчанию будет работать только в том случае, если на вашем компьютере установлены Exchange Server 2000/2003 или Internet Information Server с настроенной службой SMTP, поскольку физически сообщение просто будет помещено в каталог C:\Inetpub\mailroot\Pickup (по умолчанию), откуда его должна забрать служба Exchange Server или IIS. Однако есть и более удобный способ отправки сообщений через любой почтовый сервер, который поддерживает протокол SMTP. Для этого перед вызовом метода `Send()` мы должны настроить параметры отправки:

```
oMyMail.Configuration.Fields.Item _
  ("http://schemas.microsoft.com/cdo/configuration/sendusing") = 2
```

Здесь `"http://schemas.microsoft.com/cdo/configuration/sendusing"` — это название поля, и оно должно быть указано точно. По умолчанию значение этого поля равно 1, что означает использование каталога Pickup.

Указать почтовый сервер можно так:

```
omyMail.Configuration.Fields.Item _
  ("http://schemas.microsoft.com/cdo/configuration/smtpserver") = _
   "smtp.YourServer.com"
```

Настройка режима аутентификации производится при помощи того же объекта `CDO.Configuration`:

```
oMyMail.Configuration.Fields.Item _
("http://schemas.microsoft.com/cdo/configuration/smtpauthenticate") = 1
```

Значение 1 означает, что используется базовая аутентификация, значение 0 — без аутентификации (анонимно), значение 2 — аутентификация NTLM.

Имя пользователя и пароль можно передать точно так же:

```
oMyMail.Configuration.Fields.Item _
  ("http://schemas.microsoft.com/cdo/configuration/sendusername") = _
  "YourLogin@YourDomain.com"
oMyMail.Configuration.Fields.Item _
  ("http://schemas.microsoft.com/cdo/configuration/sendpassword") = _
  "Password"
```

Иногда необходимо также определить использование специфического порта (отличного от 25), будет или нет использоваться SSL и время тайм-аута:

```
oMyMail.Configuration.Fields.Item _
("http://schemas.microsoft.com/cdo/configuration/smtpserverport") = 2525

oMyMail.Configuration.Fields.Item _
("http://schemas.microsoft.com/cdo/configuration/smtpusessl") = False

oMyMail.Configuration.Fields.Item _
("http://schemas.microsoft.com/cdo/configuration/smtpconnectiontimeout") =
60
```

После любых изменений, вносимых в конфигурацию, их вначале надо сохранить:

```
oMyMail.Configuration.Fields.Update
```

и только после вызывать метод `Send()`:

```
oMyMail.Send
```

Если возникает проблема с кодировками (обычно, если в системе установлен русский язык, не возникает), но можно еще перед отправкой добавить строку вида:

```
oMyMail.TextBodyPart.Charset = "windows-1251"
```

К сожалению, эта библиотека работает только с протоколом SMTP и каталогом Pickup на диске. Она не умеет работать ни с протоколом POP3, ни с IMAP4, ни с MAPI, а значит, подключиться к почтовому серверу и проверить на нем появление новых сообщений (как в нашем примере с Outlook) мы не сможем. Придется использовать другую библиотеку из набора CDO — Microsoft CDO 1.21 Library. Она умеет работать только с MAPI (т. е. с Exchange Server), зато может выполнять различные операции в почтовом ящике на Exchange Server без всяких предупреждающих сообщений.

Вначале, как всегда, нужно добавить ссылку на библиотеку Microsoft CDO 1.21 Library (меню **Tools | References**). Чтобы, например, отследить появление новых писем, нужно выполнить следующий код:

```
Dim oSession As New MAPI.Session
Dim oFolder As MAPI.Folder
Dim oMessage As MAPI.Message

'"Outlook" — имя почтового профиля. Если этот параметр не передать,
'то возникнет диалоговое окно с предложением выбрать нужный профиль.
'Имя нужного профиля можно узнать как раз из этого диалогового окна
oSession.Logon ("Outlook")
Set oFolder = oSession.Inbox
For Each oMessage In oFolder.Messages
    If oMessage.Unread = True Then Debug.Print oMessage
Next
```

Полную справку по этим объектным моделям можно прочитать в MSDN.

Задание для самостоятельной работы 13: Работа с контактами в Outlook

Ситуация:

В приложении для работы с клиентами вашего предприятия есть таблица со списком организаций, в которой хранится информация о представителях этих организаций и их координатах. Сотрудники вашего предприятия, которые осуществляют связь с клиентами, просят сделать так, чтобы эта информация стала доступной для них из Outlook.

ЗАДАНИЕ:

Напишите макрос Outlook `ImportContacts()`, который бы:

1. Создавал новую папку Outlook с элементами типа **Контакт** под именем **Контакты клиентов**.
2. Создавал в этой папке контакты для всех записей таблицы `Клиенты` базы данных C:\Program Files\Microsoft Office\OFFICE11\SAMPLES\Борей.mdb.

   Чтобы упростить задачу, достаточно поместить в создаваемые контакты только информацию о названии организации, имени представителя и телефоне.

Примечание

На практике, вполне возможно, удобнее было бы создать общую папку на сервере Exchange Server и программным образом создать в ней контакты, чтобы можно было предоставить доступ к ним сразу группе пользователей. В нашем случае для простоты мы создаем элементы контактов в локальной папке Outlook. Код VBA для создания контактов в любом случае будет идентичен.

Ответ к заданию 13

1. Запустите Outlook и нажмите в нем клавиши <Alt>+<F11>, чтобы открыть редактор кода Visual Basic.
2. В меню **Tools | References** добавьте ссылку на объектную библиотеку Microsoft ActiveX Data Objects 2.1 Library.
3. В окне **Project Explorer** щелкните правой кнопкой мыши по объекту проекта **Проект1** и в контекстном меню выберите **Insert | Module**. Будет создан новый стандартный модуль **Module1**.
4. В этом модуле создайте новую процедуру `ImportContacts()` и добавьте в нее необходимый код. Он может быть таким:

```
Public Sub ImportContacts()
    Dim oFolder As MAPIFolder
    Dim oNameSpace As NameSpace
    Dim oContact As ContactItem

    Set oNameSpace = Application.GetNamespace("MAPI")
    'Создаем папку и получаем ссылку на ее объект
    Set oFolder = oNameSpace.Folders("Личные папки").Folders. _
        Add("Контакты клиентов", olFolderContacts)

    'Создаем объект соединения
    Dim cn As New ADODB.Connection
    cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" _
& "Data Source=C:\Program Files\Microsoft
Office\OFFICE11\SAMPLES\Борей.mdb"
    cn.Open

    'Создаем объект Recordset
    Dim rs As New ADODB.Recordset
    rs.Open "Клиенты", cn

    'Проходим циклом по Recordset
    Do While rs.EOF = False
        'Создаем объект контакта
        Set oContact = Application.CreateItem(olContactItem)
        'Заполняем его свойства на основе данных из Recordset
        oContact.CompanyName = rs.Fields("Название")
        oContact.FullName = rs.Fields("ОбращатьсяК")
        oContact.BusinessTelephoneNumber = rs.Fields("Телефон")
        'Перемещаем в нашу папку и сохраняем
        oContact.Move oFolder
        oContact.Save
```

```
            'Освобождаем память
            Set oContact = Nothing
            'Сдвигаемся на одну запись в Recordset
            rs.MoveNext
        Loop

    End Sub
```