Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Тузовский Ф.А.

СЕТЕВЫЕ СЕРВИСЫ С ИСПОЛЬЗОВАНИЕМ КПК И СОТОВЫХ ТЕЛЕФОНОВ

Учебно-методическое пособие

Издательство
Томского политехнического университета
2008

УДК 004.482.7
ББК 32.973.23-018
Т

Тузовский Ф.А.
Т Сетевые сервисы с использованием КПК и сотовых телефонов / Ф.А. Тузовский. – Томск: Изд-во Томского политехнического университета, 2008. – 150 с.

В данном учебном пособии рассматриваются современные мобильные вычислительные устройства и построение на их основе мобильных сетевых сервисов. Поясняются основные платформы выполнения и разработки мобильного программного обеспечения. Подробно рассматривается разработка трех видов мобильных приложений с использованием среды разработки Microsoft Visual Studio.Net: локальных мобильных приложений, мобильных приложений, использующих web-сервисы и мобильных web-приложений.

Пособие подготовлено на кафедре «Оптимизации систем управления» ТПУ и предназначено для студентов направления 230100 «Информатика и вычислительная техника», магистерской программы 230113 «Сети ЭВМ и телекоммуникации», изучавших дисциплины «Высокоуровневые методы информатики и программирования» и «Проектирование Интернет приложений».

УДК 004.482.7
ББК 32.973.23-018

Рекомендовано к печати Редакционно-издательским советом
Томского политехнического университета

Рецензенты

Профессор Томского университета систем управления и радиоэлектроники; доктор технических наук; зав. кафедрой «Хххххххх»
Х.Х. Хххххх

Профессор кафедры «Ххххххххх» Радиофизического факультета Томского государственного универсиета; доктор технических наук
Х.Х. Хххххххх

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современное общество постепенно переходит от электронного бизнеса к мобильному. И если электронные технологии обеспечивают доступность и полноту информации, то мобильные технологии в первую очередь ставят своей задачей обеспечить своевременность информации и ее релевантность. Мобильные технологии в настоящее время активно развиваются. При этом можно выделить три тенденции в этом процессе. Во-первых, в мобильные телефоны, которые приобрели большую популярность у пользователей, добавляется все больше возможностей выполнять вычисления (смартфоны), что позволяют использовать их не только для голосовой связи, но и в качестве небольших компьютеров. Во-вторых, миниатюризация компьютеров привела к созданию карманных персональных компьютеров (КПК), которые пользователь может носить с собой и в автономном режиме выполнять нужные ему программы. И, в-третьих, активно развиваются технологии и инфраструктуры беспроводной связи, которые позволяют обмениваться данными между карманными персональными компьютерами и стационарными информационно вычислительными сетями, как в рамках локальных (например, технология Wi-Fi), так и глобальной (технологии GPRS, UMTS, WiMax) компьютерных сетей. Все это происходит на фоне снижения стоимости, а значить и доступности, таких мобильных вычислительных устройств.

Такое быстрое развитие мобильных устройств и беспроводных технологий связи создает возможность предоставления их пользователям информационных и вычислительных услуг в том месте, где имеется потребность в их результатах. Появляется возможность реализовать предоставление информационных сервисов любым пользователям, в любое время и в любом месте.

В связи с развитием мобильного бизнеса и сетевых сервисов актуальным становится обучение специалистов по информационным технологиям проектированию разработке сетевых сервисов с использованием мобильных приложений.

Целью данного пособия и является систематизация, описание и обучение студентов разработке мобильных приложений. В качестве платформы разработки таких приложений в пособии использована платформа Microsoft .Net Framework и среда разработки Microsoft Visual Studio.

1. МОБИЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ УСТРОЙСТВА

Под мобильными вычислительными устройствами (МВУ) понимаются легкие, небольшого размера (карманные) устройства, которые могут автономно выполнять вычисления и подключаться к локальным и глобальным компьютерным сетям с помощью беспроводных технологий.

1.1. Категории мобильных устройств

Из всего разнообразия МВУ можно выделить четыре основные категории:

- карманные персональные компьютеры (КПК);
- коммуникаторы;
- смартфоны;
- сотовые телефоны.

Сотовым телефоном называется портативное устройство, основным назначением которого является предоставление услуг голосовой связи посредством сотовой сети. Сотовые телефоны управляются операционной системой, которая не предназначена для запуска прикладных программ (приложений).

Карманный персональный компьютер (КПК) это портативное устройство, на котором предустановленна операционная система, позволяющая устанавливать и выполнять прикладные программы.

Следует отметить, что современные сотовые телефоны также позволяют выполнять прикладные программы (приложения). Разница состоит в том, что в случае с сотовым телефоном, можно говорить лишь о запуске программ с ограниченной функциональностью, запущенных в специально выделенной изолированной среде (так называемой «песочнице»). В случае со смартфонами и коммуникаторами функциональные возможности программ практически не ограничены, например, в платформе Android, при желании, можно заменить даже программный модуль сотового телефона.

Если разделение на КПК и телефоны является очевидным, то различие между смартфонами и коммуникаторами медленно уменьшается. Само же разделение произошло исходя из того, на какой основе возникли те или иные устройства:

- Коммуникаторы – это КПК, в которые была добавлена поддержка сотовых сетей. Примером коммуникаторов могут служить устройства под управлением операционной системы Windows Mobile.

- Смартфонами – это сотовые телефоны с операционной системой, приспособленной для запуска сторонних приложений. Классическим примером смартфонов являются устройства на основе операционной системы Symbian OS.

На рис. 1.1 показано графически, как соотносятся эти категории мобильных устройств:

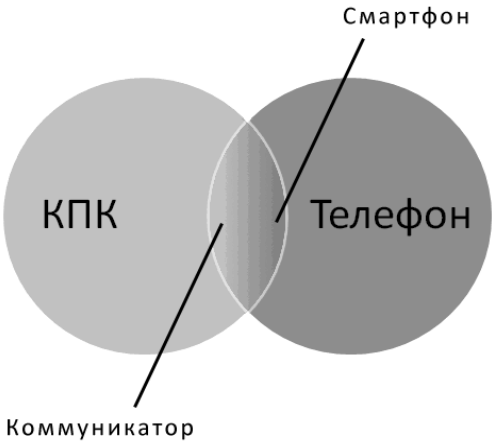

Рис. 1.1. Категории мобильных устройств

Основным фактором, который в настоящий момент позволяет разделить коммуникаторы и смартфоны является различие в принципах ввода данных: стандартным средством ввода данных для КПК и, соответственно, коммуникаторов, является сенсорный экран. В смартфонах сенсорных экранов не используются, а исторически применяется клавиатурный ввод в качестве основного.

Однако, даже эта разница медленно, но верно исчезает: появившийся в 2007 году Apple iPhone, который на момент выпуска не поддерживал возможности установки дополнительных программ и, соответственно, мог быть отнесен к категории «телефоны», имел поддержку сенсорного ввода. Популярность этого телефона привела к тому, что крупнейшие компании, создающие смартфоны, такие как Nokia и Research In Motion, принялись за внедрение сенсорного ввода для своих платформ.

1.2. Технические характеристики мобильных устройств

1.2.1. Конструкция мобильных устройств

Чаще всего, конструкцию мобильного устройства называют «форм-фактором». Под форм-фактором понимается совокупность элементов для ввода и вывода информации. В качестве примера, можно рассмотреть из каких элементов состоит коммуникатор Treo 600, приведенный на рис. 1.2 и 1.3.

Рис. 1.2. Внешний вид коммуникатора Treo 600

1) индикатор статуса;
2) динамик коммуникатора;
3) цветной экран;
4) кнопка ответа;
5) четырехпозиционная кнопка навигации;
6) кнопка включения / выключения коммуникатора и завершения разговора;
7) кнопка доступа к функциям телефона;
8) кнопка вызова стандартного приложения (календаря);
9) кнопка вызова приложения сообщений;
10) кнопка выхода в главное меню;
11) QWERTY-клавиатура;
12) клавиатура набора номера;
13) стилус;
14) зеркало для автопортрета;
15) цифровая камера;
16) внешний динамик;
17) слот для SIM-карты;
18) съемный аккумулятор;
19) кнопка регулирования громкости;
20) настраиваемая кнопка;
21) слот расширения Secure Digital;
22) отключение звука звонка;
23) гнездо для аудио-гарнитуры;
24) мульти-коннектор;
25) микрофон.

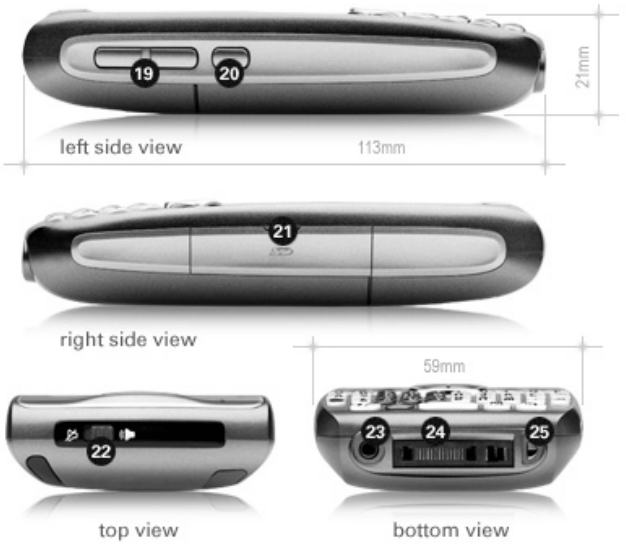

Рис. 1.3. Конструкция коммуникатора Treo 600

Если устройство, например, такое, как приведенный на рисунках выше коммуникатор, не имеет никаких подвижных частей, то его обычно называют «моноблоком». Так, коммуникатор Treo 600 имеет форм-фактор «моноблок с qwerty-клавиатурой». Впрочем, приведенный пример является далеко не единственным: среди смартфонов и коммуникаторов встречаются и «раскладушки» – когда аппарат может раскладываться на две части, «слайдеры» (причем как вертикальные, так и горизонтальные) – когда одна часть устройства сдвигается относительно другой (рис. 1.4).

Рис. 1.4. Горизонтальный «слайдер» с QWERTY-клавиатурой

Все мобильные устройства включают встроенный компьютер со всеми присущими ему составными частями, такими как, процессор и оперативная память.

1.2.2. Процессоры мобильных устройств

Ключевой процессорной архитектурой, использующейся в мобильных устройствах, стала архитектура Advanced RISC Machine (ARM). В настоящий момент, по данной архитектуре производится около 75 % из всех встраиваемых 32-битных RISC-процессоров, что делает ее одной из наиболее широко используемых архитектур в мире. Необходимо отметить, что процессоры ARM используются не только в портативных устройствах, но и в компьютерной периферии – от жестких дисков до маршрутизаторов (роутеров).

Разработкой данной архитектуры занимается компания ARM Limited. В ее задачи входит создание научных разработок и продажа лицензий сторонним фирмам. Такие процессоры, как Intel XScale или Texas Instruments OMAP являются реализациями архитектуры ARM.

В настоящее время разработано шесть основных семейств процессоров: ARM7, ARM9, ARM9E, ARM10, ARM11 и SecurCore. Также совместно с компанией Intel разработаны семейства XScale и StrongARM.

Как дополнение к ARM-архитектуре могут интегрироваться несколько расширений:

- Набор инструкций Thumb, позволяющи
инструкций, пользуясь при этом в
разрядной системы – например, доступ
адресным пространством.
- Технология SIMD (несколько данных
ляющая ускорить обработку медиа-дан
ствие аудио/видеокодеков может быть
раза.

Рис. 1.5. Процессор ARM7TDMI от Atmel

- Набор инструкций DSP, интегрирующий в процессор инструкции DSP, выполняющие 16-разрядные и 32-разрядные арифметические операции для обработки сигналов в реальном времени.
- Технология Jazelle, позволяющая процессору выполнять Java-код на аппаратном уровне.
- Технология Intelligent Energy Manager (IEM), объединяющая аппаратные и программные компоненты, которые совместно выполняют динамическое распределение питания.
- Технология TrustZone, предназначенная для защиты памяти устройства от критических сбоев, вызываемых вирусами и неправильной работе программного обеспечения.

Аппаратная платформа

Как уже можно было понять из описания платформы ARM, процессор в портативных устройствах отличается от процессоров настольных персональных компьютеров (ПК) тем, что состоит из большого количества различных модулей, отвечающих не только за вычисление. Для примера можно рассмотреть чипсет MSM7200 компании Qualcomm, под управлением которого работает большинство современных коммуникаторов под управлением ОС Windows Mobile. В этом чипе два ARM процессора: 400 МГц ARM11 для обработки приложений и 279 МГц ARM9 для работы модема, каждый из которых имеет свой цифровой сигнальный процессор (QDSP5000 и QDSP4000). Встроенный графический процессор ATi Imageon 2300 позволяет работать с 2D и 3D графикой, проигрывать видео встроенным MPEG4 декодером и захватывать фото и видео с помощью JPEG кодека. Контроллер памяти позволяет работать с NAND и SDRAM.

1.2.3. Оперативная память мобильных устройств

Используемую в мобильных устройствах память можно разделить на два типа: Random access memory (RAM) и Read-only memory (ROM).

Память с произвольной выборкой (Random access memory, RAM) представляет собой один из видов памяти, позволяющий в любой момент времени получить доступ к любой ячейке по её адресу на чтение или запись. RAM подразделяется на статическую и динамическую. В статических ОЗУ запоминающий элемент представляет собой триггер, что позволяет считывание информации без её потери. В динамических ОЗУ элементом памяти является ёмкость (например, входная ёмкость полевого транзистора), что требует восстановления записанной информации в процессе её хранения и использования. Это усложняет применение ОЗУ динамического типа, но позволяет реализовать больший объём памяти. В современных динамических ОЗУ имеются встроенные системы синхронизации и регенерации, поэтому по внешним сигналам управления они не отличаются от статических.

Можно выделить следующие типы RAM

- Полупроводниковая статическая (SRAM) память – ячейки представляют собой полупроводниковые триггеры. Достоинства – небольшое энергопотребление, высокое быстродействие. Отсутствие необходимости производить «регенерацию». Недостатки – малый объём, высокая стоимость. В настоящее время широко используется в качестве кэш-памяти процессоров.
- Полупроводниковая динамическая (DRAM) память – каждая ячейка представляет собой конденсатор на основе перехода

КМОП-транзистора. Достоинства – низкая стоимость, большой объём. Недостатки – необходимость периодического считывания и перезаписи каждой ячейки, т. н. «регенерация», и, как следствие, понижение быстродействия, большое энергопотребление. Процесс регенерации реализуется специальным контроллером, установленным на материнской плате или в центральном процессоре. DRAM обычно используется в качестве оперативной памяти, в т. ч. и мобильных устройств.

Постоянное запоминающее устройство (Read-only memory, ROM) это второй существующий вид памяти. Особенностью его является то, что данные, хранящиеся в ROM не могут быть изменены (по крайней мере, это невозможно выполнить быстро).

В строгом смысле этого слова, термин ROM может относиться только к самому старому типу твердотельных ПЗУ, известному как mask (масочный) ROM, которые производятся с заранее введенными данными и в принципе не может быть изменены. Однако, в более современных типах ПЗУ, данные могут быть стерты и перепрограммированы заново множество раз. Термин ROM (т. е. память, предназначенная только для чтения) применяется к ним потому, что процесс перепрограммирования предполагается нерегулярным, относительно медленным и, зачастую не позволяющим произвольные выборки на запись для конкретных ячеек памяти. Несмотря на простоту масочного ПЗУ, возможности по масштабированию и перепрограммированию на лету делает современные типы ROM более гибкими и недорогими в использовании, что привело к тому, что в настоящее время масочное ПЗУ в устройствах практически не используется.

Когда речь идет о мобильных устройствах, то под термином ROM подразумевается флеш-память. Флеш-память (Flash-Memory) – разновидность твердотельной полупроводниковой энергонезависимой перезаписываемой памяти. Ее особенностью является то, что она может быть прочитана сколько угодно раз, но количество перезаписей такой памяти ограничено. В настоящее время выделяют два основных типа флеш-памяти: NOR и NAND. Эти архитектуры в настоящее время существуют параллельно и не конкурируют друг с другом, поскольку находят применение в разных областях хранения данных.

Применение RAM и ROM в мобильных устройствах

Многие современные мобильные устройства применяют оба типа памяти и это обусловлено преимуществами и недостатками каждого из них. Так RAM, являясь очень быстрым типом памяти, потребляющим достаточно много энергии, в основном используется в качестве опера-

тивной памяти. ROM же намного медленнее, но гораздо более экономично расходует энергию, т. к. не нуждается в постоянном питании для хранения данных. Соответственно, в качестве носителя информации в настоящее время применяется ROM. Впрочем, этот принцип не является догматическим: так в устройствах Windows Mobile вплоть до 5-й версии вся используемая память была энергозависимой, что позволяло операционной системе динамически выделять необходимое ей количество оперативной памяти и мгновенно запускать находящиеся в RAM. Обратной стороной такого подхода было то, что при разрядке батарей, вся информация, в т. ч. контакты, установленные программ и т. д. стиралась.

Применение двух основных типов флеш-памяти: NAND и NOR тоже обуславливается особенностями их работы. Адресное пространство NOR-памяти позволяет работать с отдельными байтами или словами (2 байта). В NAND ячейки группируются в небольшие блоки (по аналогии с кластером жесткого диска). Из этого следует, что при последовательном чтении и записи преимущество по скорости будет у NAND. Однако с другой стороны NAND значительно проигрывает в операциях с произвольным доступом и не позволяет напрямую работать с байтами информации.

1.3. Коммуникационные возможности

Возникновение интернета превратило персональный компьютер из автономного устройства, которое может работать исключительно с программами и данными в его памяти, в участника локальных и глобальной компьютерной сети, которому доступны совершенно новые объемы данных и возможности.

Рис. 1.6. Ericsson R380

В конце 90-х годов прошлого века на рынке появились первые гибриды карманных персональных компьютеров и телефонов – такие Handspring Visorphone или Ericsson R380 (рис. 1.6). Чтобы дать пред-

ставление о том, какой путь за это время прошли мобильные устройства необходимо отметить, что первое из указанных устройств было не коммуникатором, а отдельной приставкой, подключавшейся к КПК на основе PalmOS; а второе, хоть и работало на основе ОС EPOC, не позволяло установку стороннего программного обеспечения.

Современные технологии по передаче данных рассмотрены ниже.

1.3.1. Стандарт GSM

Самым популярным стандартом сотовой связи в мире является стандарт GSM (Global System for Mobile Communications). Согласно данным Ассоциации GSM ему принадлежат 82 % рынка глобальной связи, более 3 миллиардов людей в 212 странах используют именно его. Необходимо отметить, что существуют мобильные устройства, которые поддерживают другие стандарты сотовой связи, такие как американский CDMA или японский FOMA. Стандарт GSM является самым популярным и наиболее распространенным стандартом сотовой связи в России и мире.

Поколения мобильных сетевых технологий

Всю историю развития мобильных сетевых технологий, которая берёт свое начало во второй половине прошлого века, можно разделить на 3 поколения, представленные в табл. 1.1.

Таблица 1.1

Поколения мобильных сетевых технологий

Поколение	1G	2G	2.5G	3G
Начало разработок	1970	1980	1985	1990
Реализация	1984	1991	1999	2002
Сервисы	аналоговый стандарт, синхронная передача данных со скоростью до 9,6 кбит/с	цифровой стандарт, поддержка коротких сообщений (SMS)	большая ёмкость, пакетная передача данных	ещё большая ёмкость, скорости до 2 Мбит/с
Стандарты	AMPS, TACS, NMT и др.	TDMA, CDMA, GSM, PDC	GPRS, EDGE, 1xRTT	WCDMA, CDMA2000, UMTS
Ширина канала	1,9 кбит/с	14,4 кбит/с	384 кбит/с	2 Мбит/с

Первое поколение (1G)

Европейским стандартом сотовой связи первого поколения стал NMT (Nordic Mobile Telephone system). Его окончательные спецификации были приняты в 1978 году пятью скандинавскими странами (Данией, Финляндией, Исландией, Норвегией и Швецией). Стандарт NMT является стандартом аналоговой сотовой связи и работает в диапазоне частот 453,0…457,5 МГц, используя до 180 каналов связи по 25 кГц каждый. Радиус действия одной базовой станции достигает 5…25 км в зависимости от нагрузки на каждую из них. В 1983 году была разработана модернизированная версия NMT-900 (первая условно называлась NMT-450), работавшая на частоте 900 МГц. Выход обновлённого стандарта позволил уменьшить размеры телефонных аппаратов, а также добавить несколько новых сервисов. Тем не менее, спустя некоторое время NMT был заменен на более прогрессивные цифровые стандарты. Вполне естественно, что первое поколение сотовой связи не смогло с ними конкурировать, т. к. несмотря на то, что качество аналоговой беспроводной связи в целом было удовлетворительным, разговор можно было легко перехватить и расшифровать.

Второе поколение (2G)

Принципиально новым подходом к передаче информации (в частности, голоса) отличалось второе поколение мобильных коммуникаций – на этот раз в его основу легли цифровые стандарты. В Европе на смену NMT пришел стандарт GSM, который применяется по настоящее время. Цифровой стандарт предполагает, что голос человека теперь проходил оцифровку (кодирование), то есть, по каналу связи, как и в 1G-стандарте, передавалась модулированная несущая частота, но уже не аналоговым сигналом, а цифровым кодом.

Внедрение цифрового стандарта привело к возникновению новых сервисов, самым известным из которых, пожалуй, можно назвать сервис обмена короткими сообщениями (SMS). Одним из существенных недостатков сетей сотовой связи стандарта GSM являлась низкая скорость передачи данных – максимум 9.6 кбит/сек. Организация этого процесса тоже была далека от совершенства – для передачи данных абоненту выделялся один голосовой канал, а биллинг осуществлялся исходя из времени соединения.

Второе с половиной поколение (2.5G)

Для расширения возможностей передачи данных посредством существующих GSM-сетей была разработана услуга пакетной передачи данных по радиоканалу GPRS (General Packet Radio Service). Для передачи данных в GPRS начали использоваться одновременно многие каналы связи в паузах между передачей речи, что позволило увеличить

скорость и производить учет и оплату услуг (биллинг) пропорционально объему переданной информации. В связи с минимальной приоритетностью GPRS пакетов, при звонке соединение по GPRS временно приостанавливается или обрывается.

Именно появление GPRS позволило реализовать мобильный Интернет, предоставив возможность получать доступ к глобальной сети с сотовых телефонов, смартфонов и коммуникаторов. Тем не менее, скорость передачи данных при использовании GPRS оставляет желать лучшего. Официально максимальный его предел равен 115 кбит/сек, а в реальности обмен информацией производится не быстрее, чем на скорости 40…50 кбит/с, что в два раза меньше теоретического максимума. По сегодняшним меркам такой пропускной способности не хватит для нормальной работы в сети Интернет.

В качестве решения этой проблемы был разработана технология EDGE (Enhanced Data rates for GSM Evolution), которую иногда относят к отдельному поколению – 2.75G. EDGE является расширением GPRS и не может существовать отдельно от GPRS. Главное различие между GPRS и EDGE – в использовании иной модуляционной схемы на физическом уровне, что позволило достичь пропускной способности, примерно втрое больше, чем в технологии GPRS.

Третье поколение (3G)

Термин 3G используется для описания сервисов мобильной связи стандарта следующего поколения, которые обеспечивают более высокое качество звука, а также высокоскоростной доступ в интернет и мультимедийные сервисы. Мобильные сети третьего поколения отличаются от сетей второго поколения гораздо большей скоростью передачи данных, а также более широким набором и высоким качеством предоставляемых услуг.

Хотя существует много различных интерпретаций того, что представляет собой 3G, единственным определением, принимаемым универсально, является определение, опубликованное Международным Институтом Электросвязи (ITU). ITU, работающий с промышленными организациями по всему миру, определяет и утверждает технические требования и стандарты, а также правила использования спектра для систем 3G в рамках программы IMT-2000 (International Mobile Telecommunications-2000). IMT-2000 – это рекомендации, разработанные Международным Институтом Электросвязи (ITU), касающиеся вопросов использования частотного спектра и технических особенностей для всего семейства стандартов 3-го поколения. Рекомендации описывают пути эволюции существующих в мире стандартов 2-го поколения в стандарты 3-го поколения. ITU требует, чтобы сети IMT-2000 (3G),

помимо прочих свойств, обеспечивали улучшенную ёмкость системы и эффективность использования спектра для систем 2G и поддерживали сервисы передачи данных со скоростями – минимум 144 кбит/с, при использовании в мобильном режиме (не в помещениях), и максимум 2 Мбита/с, в не мобильных условиях (в помещениях).

Основываясь на этих требованиях, в 1999 году ITU одобрил пять радиоинтерфейсов для стандартов IMT-2000, как часть рекомендаций ITU-R M.1457. Ключевыми стали два стандарта 3G: UMTS (Universal Mobile Telecommunications Systems – универсальная мобильная телекоммуникационная система), поддерживаемая европейскими странами, и CDMA 2000 (Code Division Multiple Access – множественный доступ с кодовым разделением каналов), сторонниками которой традиционно являются азиатские страны и США. В принципе эти две технологии предполагают два различных подхода к организации сетей 3G: революционный (UMTS) и эволюционный (разновидности CDMA – CDMA2000, CDMA2000 IX, CDMA2000 IX EvDo). Эволюционный путь подразумевает сохранение частот и постепенный переход к новым технологиям, путём наращивания технических мощностей оператора. UMTS – совершенно новый стандарт, в то время как разновидности CDMA, предложенные для 3G, являются развитием уже эксплуатирующейся в мире технологии второго поколения cdmaOne (IS-95).

Всего существует три основных стандарта 3G: UMTS (Universal Mobile Telecommunications Service), CDMA2000 и WCDMA (Wide CDMA). Все они настроены на пакетную передачу данных и, соответственно, на работу с цифровыми компьютерными сетями, включая Интернет. Скорость передачи данных в новом поколении стандартов может достигать 2,4 Мбит/с. Это позволит поднять качество звука, а также добавить такой сервис, как видеозвонок, о котором, вероятно, слышали уже многие. Мобильный Интернет теперь станет доступнее и значительно быстрее.

В настоящее время сети 3G уже работают в Азии, США, в то время как в Европе существует пока только в тестовых вариантах. Наиболее впечатляющих успехов в области 3G на мировом фоне добилась Япония. По состоянию на 2008 г. в Японии работают четыре оператора, которые предоставляют услуги третьего поколения, – это NTT DoCoMo, KDDI AU, SoftBank и EMobile. Используя CDMA2000 и WCDMA технологии, первые всемирные коммерческие 3G сети уже обслуживают миллионы абонентов. К концу октября 2002 года, KDDI подключил 3,9 миллиона абонентов CDMA2000, NTT DoCoMo – 149,000 абонентов FOMA (WCDMA). Также к концу октября общее количество абонентов в Корее составило более чем 15 миллионов абонентов CDMA2000.

По данным на 16 декабря 2002 года в мире запущено 32 сети третьего поколения в 16 странах.

1.3.2. Технология Wi-Fi

В основе WLAN-технологий лежит принцип высокочастотной радиосвязи между узлами сети. В качестве узла сети может выступать как отдельный компьютер, ноутбук или КПК, так специальный устройство «точка доступа» («Access Point») обеспечивающее доступ к кабельному сегменту сети Ethernet, к Интернету или другому компьютеру.

Специальные стандарты для WLAN-сетей разрабатываются Институтом инженеров по электротехнике и электронике (Institute of Electrical and Electronics Engineers) более известного под аббревиатурой IEEE. Первый стандарт IEEE 802.11 для беспроводных локальных сетей был принят в 1997 году. Он подразумевал работу оборудования на частоте 2.4 ГГц со скоростями 1 и 2 Мб/с. Стандарт разрабатывался в течение 7 лет и поэтому ко времени принятия уже не мог соответствовать выросшим потребностям.

Новый расширенный вариант стандарта, названный 802.11b (802.11 High Rate), был принят в 1999 году. С его принятием стала возможной работа беспроводных сетей на скоростях до 11 Мб/с, что было сопоставимо по скорости с обычными сетями Ethernet. Такая скорость позволила существенно расширить область применения беспроводных сетей и поднять уровень задач, для которых стало возможным использование WLAN.

В том же 1999 году была создана независимая международная организация Wi-Fi Alliance (Wi-Fi – сокращение от Wireless Fidelity), занимающаяся сертификацией на совместимость WLAN-устройств от различных производителей. Эта организация объединяет практически всех ведущих производителей Intel, IBM, Cisco, HP, Dell и других. В настоящий момент в нее входят более 200 компаний, и уже более 1500 устройств получили сертификат Wi-Fi с момента начала сертификации в марте 2000 года.

Торговая марка Wi-Fi гарантирует совместимость оборудования от разных производителей. Первоначально в ноутбуках использовались адаптеры стандарта 802.11b, поэтому логотип WiFi часто ассоциировался именно с этим стандартом. В настоящее время под Wi-Fi понимается любой из стандартов 802.11: a, b или g.

1.3.3. Стандарты передачи данных IEEE 802.11

Существует несколько разновидностей WLAN-сетей, которые различаются схемой организации сигнала, скоростями передачи данных, радиусом охвата сети, а также характеристиками радиопередатчиков

и приемных устройств. Наибольшее распространение получили беспроводные сети стандарта IEEE 802.11b, IEEE 802.11g и IEEE 802.11a. Их сравнение представлено в табл. 1.2.

Таблица 1.2

Стандарты IEEE 802.11

Характеристики	Спецификации		
	IEEE 802.11a	IEEE 802.11b	IEEE 802.11g
Скорость передачи данных	до 54 Мбит/с	11 Мбит/с	до 54 Мбит/с
Расстояние и скорость передачи данных	В закрытых помещениях: 12 м (54 Мбит/с), 91 м (6 Мбит/с) В открытых помещениях в пределах прямой видимости: 30 м (54 Мбит/с), 305 м (6 Мбит/с)	В закрытых помещениях: 30 м (11 Мбит/с), 91 м (1 Мбит/с) В открытых помещениях в пределах прямой видимости: 120 м (11 Мбит/с), 460 м (1 Мбит/с)	В закрытых помещениях: 30 м (54 Мбит/с), 91 м (1 Мбит/с) В открытых помещениях в пределах прямой видимости: 120 м (54 Мбит/с), 460 м (1 Мбит/с)
Рабочая частота	5 ГГц (5,15…5,350 ГГц и 5,725…5,825 ГГц)	2,4 ГГц (2,4…2,4835 ГГц)	2,4 ГГц (2,4…2,4835 ГГц)

Первыми в 1999 г. были утверждены спецификации 802.11a и 802.11b, однако наибольшее распространение получили устройства, выполненные по стандарту 802.11b.

В стандарте 802.11b применяется метод широкополосной модуляции с прямым расширением спектра – DSSS (Direct Sequence Spread Spectrum). Весь рабочий диапазон делится на 14 каналов, разнесенных на 25 МГц для исключения взаимных помех. Данные передаются по одному из этих каналов без переключения на другие. Возможно одновременное использование всего 3 каналов. Скорость передачи данных может автоматически меняться в зависимости от уровня помех и расстояния между передатчиком и приемником.

Стандарт IEEE 802.11b обеспечивает максимальную теоретическую скорость передачи 11 Мбит/с, что сравнимо с обычной кабельной сетью 10 BaseT Ethernet. Однако, такая скорость возможна лишь при условии, что в данный момент только одно WLAN-устройство осуществляет передачу. При увеличении числа пользователей полоса пропускания делится на всех и скорость работы падает. Несмотря на ратификацию

стандарта 802.11a в 1999 году, он реально начал применяться только с 2001 года. Данный стандарт используется, в основном, в США и Японии. В России и в Европе он не получил широкого распространения.

В стандарте 802.11a используется OFDM схема модуляции сигнала – мультиплексирование с разделением по ортогональным частотам (Orthogonal Frequency Division Multiplexing). Основной поток данных разделяется на ряд параллельных подпотоков с относительно низкой скоростью передачи, и далее для их модуляции используется соответствующее число несущих. В стандарте определены три обязательные скорости передачи данных (6, 12 и 24 Мбит/с) и пять дополнительных (9, 18, 24, 48 и 54 Мбит/с). Также имеется возможность одновременного использования двух каналов, что обеспечивает увеличение скорости вдвое.

Стандарт 802.11g окончательно был ратифицирован в июне 2003 г. Он является дальнейшей разработкой спецификации IEEE 802.11b и осуществляет передачу данных в том же частотном диапазоне. Основным преимуществом этого стандарта является увеличенная пропускная способность – скорость передачи данных составляет до 54 Мбит/с по сравнению с 11 Мбит/с у 802.11b. Как и IEEE 802.11b, новая спецификация предусматривает использование диапазона 2,4 ГГц, но для увеличения скорости применена таже схема модуляции сигнала – что и в 802.11a – ортогональное частотное мультиплексирование (OFDM).

Особенностью данного стандарта является совместимость с 802.11b. Например, адаптеры 802.11b могут работать в сетях 802.11g (но при этом не быстрее 11 Мбит/с), а адаптеры 802.11g могут снижать скорость передачи данных до 11 Мбит/с для работы в старых сетях 802.11b.

Сейчас ведутся разработки нового стандарта WLAN – IEEE 802.11n. Он должен работать вдвое быстрее, чем 802.11a и 802.11g, на скорости от 100 Мбит/с до максимального значения 540 Мбит/с. Ожидается, что окончательные спецификации 802.11n будут приняты в 2006 году предположительно в ноябре.

1.3.4. Протокол Bluetooth

Bluetooth – технология беспроводной связи, созданная в 1998 году группой компаний: Ericsson, IBM, Intel, Nokia, Toshiba. В настоящее время разработки в области Bluetooth ведутся Bluetooth SIG (Special Interest Group), в которую входят также Lucent, Microsoft и многие другие.

Основное назначение Bluetooth – обеспечение экономичной (с точки зрения потребляемого тока) и дешевой радиосвязи между различными

типами электронных устройств, причем немалое значение придается компактности электронных компонентов, что дает возможность применять Bluetooth в малогабаритных устройствах размером с наручные часы.

Интерфейс Bluetooth позволяет передавать голос и данные. Для передачи данных могут быть использованы асимметричный и симметричный методы. Работающий на частоте 2.4 ГГц приемопередатчик, коим является Bluetooth-чип, позволяет в зависимости от степени мощности устанавливать связь в пределах 10 или 100 метров. Разница в расстоянии, безусловно, большая, однако соединение в пределах 10 м позволяет сохранить низкое энергопотребление, компактный размер и достаточно невысокую стоимость компонентов.

Одним из выгодных моментов в реализации этой технологии является рабочий диапазон частот – 2,45 ГГц; это так называемый нижний диапазон, обозначаемый аббревиатурой ISM (Industrial, Scientific, Medical) и использующийся для работы промышленного, научного и медицинского оборудования. Он разрешен к свободному использованию по всему миру, и для применения соответствующих устройств не требуется дополнительных лицензий и разрешений.

Всего существует три коммерческие версии протокола: 1.1, 1.2 и 2.0. Все они обратно совместимы друг с другом. В 2004 году к протоколу Bluetooth 2.0 была добавлена функция «Enhanced Data Rate» (EDR) позволившая увеличить скорость передачи данных втрое.

Классов в Bluetooth три, и отвечают они за радиус действия адаптера/модуля: class 1 (до 100 метров), class 2 (до 10 метров) и class 3 (до 1 метра). Необходимо однако учитывать, что дальность действия в случае технологии Bluetooth – понятие весьма абстрактное: стабильная работа спаренных устройств на расстоянии 10 или 100 метров (в зависимости от класса) может быть обеспечена только в идеальных условиях, которые в реальной жизни практически недостижимы. Кроме того, если одно устройство поддерживает Bluetooth class 2, а другое – class 1, то функционировать они смогут лишь на расстоянии до 10 метров.

Функциональность устройств по данному протоколу реализуется с использованием профилей Bluetooth. Профили Bluetooth это своеобразный механизм, обеспечивающий функционирование связки двух и более Bluetooth-устройств: если каждое из них поддерживает один и тот же профиль, определенный в спецификации Bluetooth, то они смогут взаимодействовать соответствующим образом. Теоретически единственным обязательным профилем, который поддерживается любым Bluetooth-модулем, является GAP (Generic Access Profile) – профиль общего доступа. Однако де-факто к нему в подавляющем большинстве случаев добавляются еще три профиля, необходимые для организации

передачи данных: профиль последовательного порта SPP (Serial Port Profile), протокол приложения определения предлагаемых сервисов SDAP (Service Discovery Application Profile) и протокол операции клиент-сервер при обмене объектами GOEP (Generic Object Exchange Profile). Помимо вышеперечисленных профилей, Bluetooth-устройство может (но отнюдь не обязано) поддерживать какие-либо из девяти основных или двенадцати дополнительных профилей.

1.3.5. Организация беспроводных сетей

Выделяют два вида организации беспроводных сетей: Ad-Hoc и Infrastructure Mode. Режим Ad-Hoc является простейшей структурой локальной сети, при которой узлы сети (ноутбуки или компьютеры) связываются напрямую друг с другом. Такая структура удобна для быстрого развертывания сетей. Для ее организации требуется минимум оборудования – каждый узел должен быть оборудован адаптером WLAN.

В режиме «Infrastructure Mode» узлы сети связаны друг с другом не напрямую, а через точку доступа (Access Point). Различают два режима взаимодействия с точками доступа – BSS (Basic Service Set) и ESS (Extended Service Set). В режиме BSS все узлы связаны между собой через одну точку доступа, которая может играть роль моста для соединения с внешней кабельной сетью. Режим ESS представляет собой объединение нескольких точек доступа, т. е. объединяет несколько сетей BSS. В этом случае точки доступа могут взаимодействовать и друг с другом. Расширенный режим удобно использовать тогда, когда необходимо объединить в одну сеть несколько пользователей или подключить нескольких проводных сетей.

Важным вопросом при организации WLAN-сетей является дальность покрытия. На этот параметр влияет сразу несколько факторов:

1. Используемая частота (чем она больше, тем меньшая дальность действия радиоволн).
2. Наличие преград между узлами сети (различные материалы поразному поглощают и отражают сигналы).
3. Режим функционирования – Infrastructure Mode) или Ad Hoc.
4. Мощность оборудования.

Если рассматривать идеальные условия, то зона покрытия с одной точкой доступа будет иметь следующий средний радиус покрытия:

- сеть стандарта IEEE 802.11a – 50 м,
- сети 802.11b и 802.11g – порядка 100 м.

За счет увеличения количества точек доступа (в режиме Infrastructure ESS) можно расширять зоны покрытия сети на всю необходимую область охвата.

1.3.6. Безопасность беспроводных сетей

Для WLAN-сетей очень актуальны вопросы безопасности и защиты передаваемых данных, так как для перехвата данных в общем случае достаточно просто оказаться в зоне действия сети.

Первоначально созданные в этой сфере технологии обладали невысокой степенью защиты, данная проблема остается актуальной и на сегодняшний день.

Для защиты передаваемых данных предусмотрены следующие методы:

- использование MAC-адресов (Media Access Control ID): у каждого адаптера есть свой, абсолютно уникальный код, установленный производителем. Эти адреса необходимо занести в списки адресов доступа у используемых для организации сети точек доступа. Все остальные WLAN-адаптеры с неправильными адресами будут исключены из сети автоматически.
- использование ключей SSID (Service Set Identifier): каждый легальный пользователь сети должен получить от администратора сети свой уникальный идентификатор сети.
- шифрование данных.

Первые два способа не обеспечивают защиты от прослушивания и перехвата пакетов данных, поэтому защитить сеть в случае перехвата данных можно только с помощью шифрования.

Изначально стандарт 802.11 предусматривал аппаратный протокол шифрования данных WEP (Wired Equivalent Privacy – защищенность, эквивалентная беспроводным сетям), основанный на алгоритме шифрования RC4. Однако, в скором времени было обнаружено, что защищенную с его помощью сеть довольно легко взломать. Ранние версии предусматривали шифрование с использованием 40-битного ключа, более поздние 64–, 128 или 256–битное. Но даже такая длина ключа в WEP не может обеспечить высокий уровень защиты сети, т. к. основная слабость данной технологии заключается в статичности ключа шифрования. Хотя при использовании данного ключа увеличивается время взлома и количество пакетов данных, которые нужно перехватить, чтобы вычислить ключ, сама возможность взлома остается. Это абсолютно неприемлемо для определенного круга серьёзных компаний и организаций.

На смену WEP была создана новая технология WPA (Wi-Fi Protected Access), разрабатывающаяся IEEE совместно с Wi-Fi Alliance. Главной особенностью новой системы безопасности является шифрование данных с динамическими изменяемыми ключами и проверка аутентификации пользователей.

В отличие от WEP здесь используется протокол целостности временных ключей TKIP (Temporal Key Integrity Protocol), который подразумевает обновление ключей перед началом каждой сессии шифрования и проверкой пакетов на принадлежность к данной сессии.

Для аутентификации пользователей используются сертификаты RADIUS (Remote Authentication Dial-In User Service – сервер RADIUS должен подтвердить право доступа). Такой метод подразумевает, главным образом, корпоративное использование.

Второй упрощенный вариант аутентификации требует предварительной установки разделяемых паролей на сетевые устройства (режим аутентификации PSK (Pre-Shared Keys)). Этот метод лучше всего применять в домашних условиях или там, где не происходит обмен важной информацией.

Изначально WPA разрабатывалась как временная технология, которая со временем она должна быть заменена новым стандартом 802.11i. Данный стандарт, с учетом всех уже существующих наработок, призван обеспечить надежное шифрование передаваемых данных, а также аутентификацию пользователей сети. В большинство уже выпущенных WiFi-устройств (точки доступа, сетевые карты) можно установить протокол WPA посредством обновления программного обеспечения.

1.4. Программные платформы

Разработка и выполнения программ на мобильных устройствах выполняется с помощью комплекса программ (операционная система, промежуточное программное обеспечение и ключевые приложения), которые совместно называются программной платформой. В настоящее время для мобильных устройств разработаны различные программные платформы. На разных типах мобильных устройств преобладающими являются разные платформы.

По исследованиям компании Gartner, в III-ем квартале 2005 года по сравнению с III-им кварталом 2004 года, рынок КПК вырос на 20,7 %, и на сегодняшний день он поделен между ОС в следующих пропорциях[1]:

- Windows Mobile на Pocket PC – 49,2 % (рост);
- BlackBerry – 25,0 % (рост);
- Palm OS – 14,9 % (сокращение);
- Symbian OS – 5,8 % (рост)
- Familiar и другие основанные на Linux – 0,7 % (стабильно);
- другие – 4,4 % (стабильно).

[1] (http://ru.wikipedia.org/wiki/Карманный_персональный_компьютер):

Исследование компании Gartner также показало, что в 2006 г. на рынке КПК преобладали устройства под управлением операционной системы Windows Mobile – 56,1 процентов.

1.4.1. Windows Mobile

Windows Mobile это операционная система и набор базовых приложений для мобильных устройств, основанная на API Microsoft Win32. Под управлением Windows Mobile могут работать как уже упоминавшиеся в данном пособии мобильные устройства, такие как смартфоны, коммуникаторы и КПК; так и портативные медиа-центры и даже бортовые компьютеры автомобилей. Визуально Windows Mobile создавался с целью походить на настольную версию Windows. Со времен первой Pocket PC 2000 вышло несколько версий ОС Windows Mobile, последней из которых на данный момент является Windows Mobile 6.1.

Пользовательский интерфейс Windows Mobile

Аналогом рабочего стола в Windows Mobile служит так называемый экран «Сегодня», изображенный на рис. 1.7. Информация на этот экран выводится набором элементов, похожих на виджеты в MacOS или приложения для боковой панели Vista – их наличие и порядок может быть изменен, могут быть установлены новые элементы. В стандартной конфигурации, на этом экране отображается текущая дата, информация о владельце, предстоящих встречах из календаря, новых сообщениях и задачах.

Рис. 1.7. Экран «Сегодня» в Windows Mobile 6

В верхней части экрана находится панель задач с кнопкой «Пуск». На панели задач отображается текущее время, настойки громкости и информация о текущем статусе сотовой связи. При открытии программы, пустое место после часов занимает иконка закрытия программы. Кнопка «Пуск» в Windows Mobile работает подобно аналогичной кнопке в настольной ОС. Она показывает избранные и недавно запускавшиеся приложения, открывать список программ, настоек и помощь.

В состав Windows Mobile входит набор приложений для работы с документами Office Mobile. Мобильная версия Microsoft Office включает в себя Word Mobile, Excel Mobile и PowerPoint Mobile. Необходимо

отметить, что функциональность мобильных и настольных версий программ пакета Office отличаются, однако специальная утилита ActiveSync – программа по синхронизации данных между мобильными устройствами и ПК, умеет конвертировать документы настольной версии Office в документы, пригодные для работы в Office Mobile. Контакты, данные календаря, почтовые сообщения и задачи могут быть синхронизованы с настольным компьютером. Также возможен доступ к почте через протоколы POP3 или IMAP4.

В состав Windows Mobile входит web браузер Internet Explorer Mobile, а Windows Media Player является стандартным проигрывателем для медиа-файлов.

Технология Internet Connection Sharing (ICS) позволяет подключаться к интернету через устройство на основе Windows Mobile посредством USB или Bluetooth.

Платформа разработки и выполнения программ .Net Compact Framework

Платформа .NET Compact Framework (.Net CF) представляет собой реализацию платформы .NET Framework для мобильных устройств. .NET CF совместим с основной, настольной версией платформы .Net Framework, но она значительно оптимизирована для устройств с ограниченными ресурсами, такими как производительность процессора, объемы памяти и емкость элементов питания, а также включает в состав библиотеки новые классы, предназначенные для разработки программ для мобильных устройств. Платформа .NET CF доступна в различных версиях ОС Windows Mobile, включая Pocket PC 2002 и 2003, Smartphone 2003 и Windows Mobile 5.0.

Среда .NET CF включает три основные компоненты:

1. Единую среду выполнения (Common Language RunTime);
2. Базовая библиотека классов (Compact Framework Class Libraries);
3. Промежуточный язык программ (Microsoft Intermediate Language).

Единая среда выполнения для .NET Compact Framework

Основываясь на принципах построения общеязыковой среды выполнения (Common Language RunTime, CLR) для .NET Framework, CLR для .NET Compact Framework (CF) была написана с нуля. Код, написанный на C# или Visual Basic.NET компилируется в промежуточный язык (MSIL), который соответствует спецификациям стандартного языкового интерфейса (CLI). MSIL-код, в свою очередь, может быть откомпилирован с помощью Just In Time (JIT) компилятора, вызываемого CLR в исходный машинный код для процессора мобильного устройства. Безопасность ти-

пов и управление памятью также выполняются CLR. Как можно догадаться, вопросы управления памятью стали одной из самых сложных задач для команды, работавшей над Microsoft .NET CF, т. к. именно эффективность при работе с памятью оказывает первостепенное влияние на скорость работы приложений и работоспособность системы в целом. Устройства под управлением Windows Mobile обычно имеют от 32 до 128 мегабайт памяти (флэш или RAM). Соответственно, пространство оперативной памяти, занимаемое .NET CF и любыми .NET-приложениями запущенными на его основе должно быть достаточно мало, при этом предоставляя поддержку большому количеству классов (хотя и не всем классам .NET Framework) приспособленных для Windows Mobile.

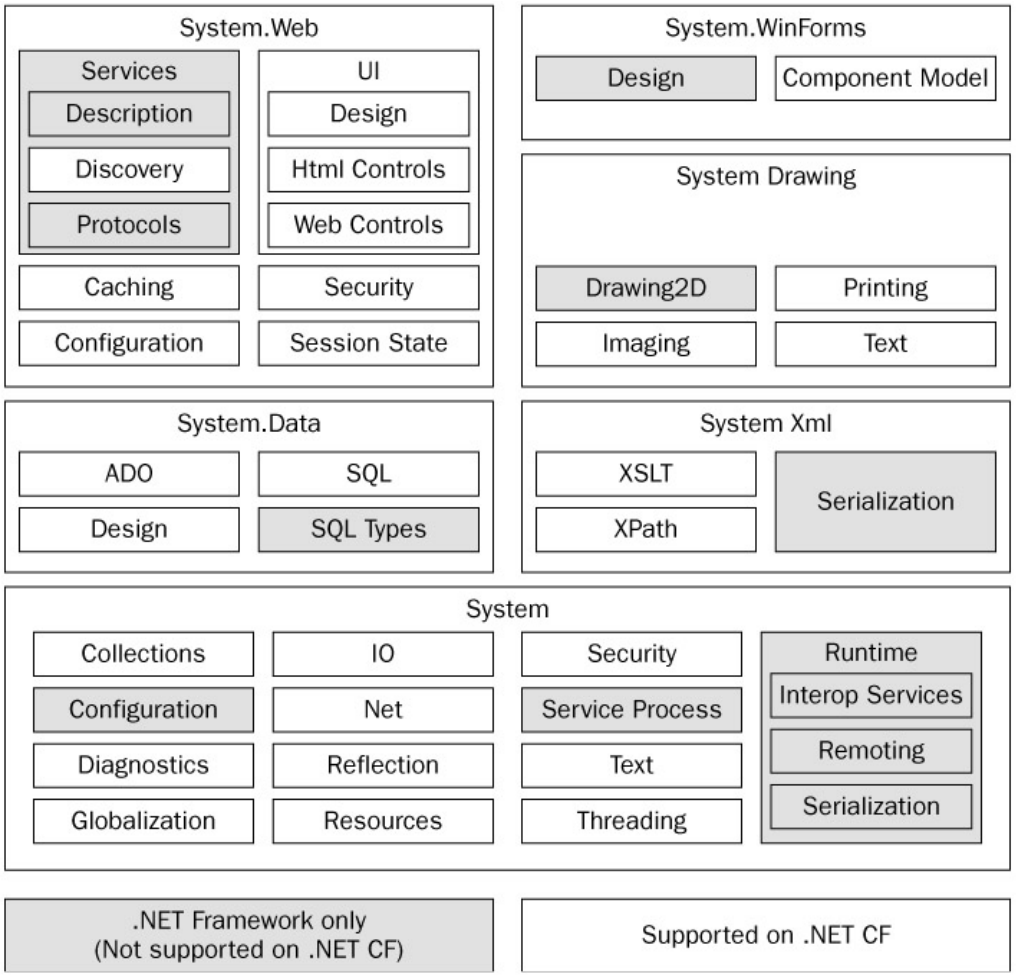

Рис. 1.8. Состав базовой библиотеки классов .Net Compact Framework

.NET Compact Framework Class Libraries

Библиотеки классов .NET CF состоят из выборки из настольных классов .NET Framework и нескольких новых классов, созданных для работы с мобильными программами и сервисами. Для того, чтобы со-

кратить размер CLR, некоторые классы были убраны, функциональность других урезана. На рис. 1.8 можно увидеть разницу между классами .NET Framework и .NET CF.

Библиотеки .NET CF имеют пространство имен, схожее с пространством имен классов .NET Framework. Разработчики, знакомые с разработкой настольных приложений .NET могут применять свой опыт по работе с классами настольных приложений для создания приложений для .NET CF. Не смотря на это, между некоторыми широко используемыми классами этих двух платформ существуют и значительные различия. Например, некоторые события, методы или свойства в Windows Forms, некоторые элементы управления, такие как `RichTextBox` и `CheckListBox`, операции drag&drop и .NET Remoting в .NET CF не поддерживаются.

1.4.2. Платформа Android

Программная платформа Android разрабатывается консорциумом, в который входит 34 компании – Open Handset Alliance. В настоящее время аппаратного обеспечения под управлением ОС Android еще нет в продаже, его выход запланирован на конец 2009 года. Однако с 2007 года компанией Google выпущен и развивается набор средств для разработки ПО. Целью SDK является предоставить инструменты и API, необходимые для начала разработки приложений на языке Java для платформы Android.

Средства разработки программного обеспечения

- Каркас приложений, позволяющий использовать много раз и замещать существующие компоненты
- Виртуальная машина Dalvik, оптимизированная для мобильных устройств
- Встроенный браузер на основе WebKit
- Оптимизированная работка графики, реализованная специальной 2D библиотекой и 3D графикой согласно спецификациям OpenGL ES 1.0
- SQLite для структурированного хранения данных
- Поддержка основных фото, аудио и видеоформатов (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM-телефония
- Bluetooth, EDGE, 3G и Wi-Fi
- Камера, GPS, компас и акселерометр

Среда разработки, включающая эмулятор устройств, инструменты для отладки, профилирование памяти и производительности, и плагин для Eclipse IDE.

Архитектура платформы Android

На рис. 1.9 показана диаграмма с описанием основных компонентов операционной системы Android. Каждый из элементов подробнее описывается ниже.

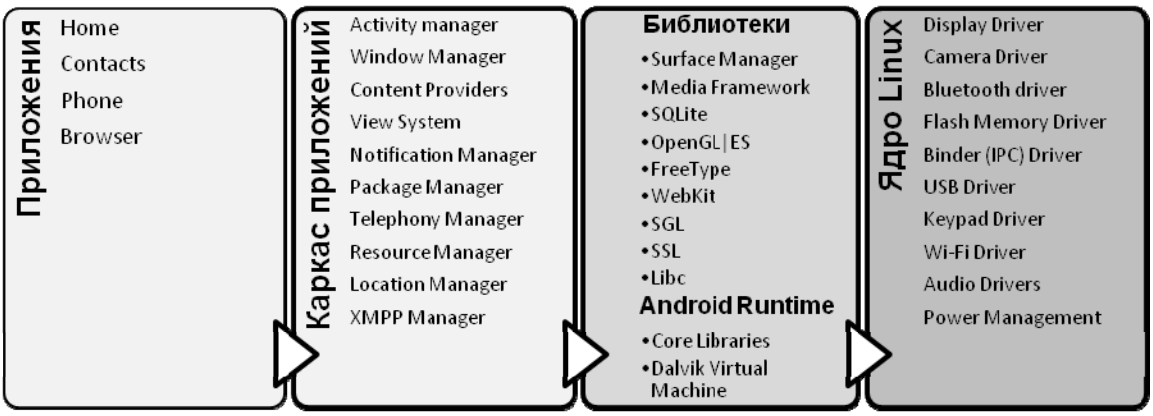

Рис. 1.9. Архитектура платформы Android

Приложения

Платформа Android будет поставляться с набором основных приложений, таких как почтовый клиент, программа по работе с SMS, календарь, карты, браузер, контакты и др. Все эти программы написаны на языке Java.

Разработчики программ на платформе Android имеют полный доступ ко всем API, доступным ключевым приложениям. Архитектура приложений создается для упрощения повторного использования компонентов, любое приложение может опубликовать свои возможности и любое другое приложение может ими воспользоваться (учитывая ограничения безопасности, накладываемые каркасом приложений). Этот же механизм позволяет пользователю заменять компоненты.

Работоспособность всех приложений обеспечивает набор сервисов и систем, включающий:

- Набор графических элементов, которые используется для построения приложений, включая списки, сетки, текстовые окна, кнопки и даже встраиваемый web-браузер.
- Контент-провайдер, который позволяет приложениям обращаться к данным из других приложений (таких как Контакты) и делиться собственными данными
- Менеджер Ресурсов, обеспечивающий доступ к ресурсам, не относящимся к коду, таким как файлы локализации, графические файлы и т. д.

- Менеджер Напоминаний, позволяющий выводить на строку статуса предупреждения
- Менеджер Процессов, управляющий жизненным циклом приложений и обеспечивающий возможность переключения между ними.

Библиотеки

Платформа Android включает в себя набор C/C++ библиотек, используемых различными компонентами системы Android. Все эти возможности доступны разработчикам через каркас приложений Android. Некоторые из них приводятся ниже:

- Системная библиотека на языке C – производная от BSD имплементация стандартной системной библиотеки языка C (libc), настроенной для работы с мобильными устройствами на основе Linux.
- Медиа-библиотеки – основанные на OpenCORE от PacketVideo. Это библиотеки обеспечивают возможность проигрывания и записи большинства наиболее популярных фото-, аудио- и видео-форматов, таких как MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG.
- Экранный менеджер – управляет доступом к подсистеме дисплея, позволяя одновременно работать 2D и 3D графике различных приложений.
- LibWebCore – современное базовое ПО для поддержки работы браузеров, обеспечивающий работу не только самого браузера Android, но и встраиваемого в приложения web обозревателя.
- SGL – встроенная 2D система графической визуализации.
- 3D библиотеки – основанные на API OpenGL ES 1.0, библиотеки могут использовать и аппаратное ускорение 3D, если оно предусмотрено, и оптимизированную программную 3D растеризацию.
- FreeType – работа с векторными и растровыми шрифтами
- SQLite – производительная и компактная реляционная база данных, доступная всем приложениям

Android Runtime

Android включает в себя набор библиотек, которые представляют большую часть функциональности ключевых библиотек языка программирования Java. Каждое приложение для Android работает в своем отдельном процессе, со собственным экземпляром виртуальной машины Dalvik. Dalvik был написан с расчетом, что несколько виртуальных машин могут одновременно эффективно работать. Dalvik выполняет файлы формата .dex, которые оптимизированы для работы с минимальным объемом оперативной памяти. Виртуальная машина основана на регистрах, и запускает классы, скомпилированные компилятором

языка Java, которые были превращены в формат .dex специальной утилитой dx из Android SDK.

Для реализации функциональности низкого уровня, такой как организация поточной обработки и низкоуровневое управление памятью, виртуальная машина Dalvik работает с ядром Linux.

Ядро Linux

Основные системные сервисы Android, такие как безопасность, управление памятью, управление процессами, сетевой стек и модель драйверов основываются на Linux версии 2.6. Ядро также работает как уровень абстракции между аппаратным обеспечением и стеком программ.

1.4.3. Java 2 Micro Edition (J2ME)

Язык программирования Java был создан в 1995 году компанией Sun Microsystems. Разработан он для того, чтобы программы написанные программистам один раз могли бы работать на разных типах мобильных устройств. В 1998 году произошло разделение языка Java на Standard Edition (J2SE), который предназначался для обычных компьютеров, Enterprise Edition (J2EE), используемый на серверах, и Micro Edition (J2ME), который и устанавливается в мобильные устройства.

Как уже говорилось в первой главе, основным отличием мобильных телефонов от смартфонов и коммуникаторов является то, что все они работают не под управлением полноценной мобильной ОС, а под управлением прошивки и установка каких-либо новых программ невозможна. Однако решение было найдено именно с помощью Java ME.

Идея Java состоит в том, что команды отдаются не напрямую процессору, а виртуальной Java-машине (JVM – Java Virtual Machine). Вместо команд процессора программа на Java представляет собой байт-код – команды, которые и должна выполнять Java-машина. Соответственно, для работы программы достаточно, чтобы в системе была установлена Java-машина. Решением поддержки приложений в сотовых телефонах стала установка в прошивки виртуальной Java ME машины под названием KVM – Kilobyte Virtual Machine.

Для программ, которые рассчитаны на Java ME, есть особое название – мидлет. Мидлеты распространяются не в виде разрозненных файлов, а в виде специальных архивов JAR – Java Archive. В нем хранятся все файлы программы: .class (они содержат байт-код), файлы ресурсов (например, картинки или звуки) и файл-манифест, описывающий программу: название, производитель, версия и другие данные.

Функционал телефонов может серьезно отличаться, поэтому для Java ME существует целый ряд различных API. Базовый API, на котором строятся все остальные, – это либо CDC (Connected Device Configuration), либо CLDC (Connected Limited Device Configuration). Для смартфонов, коммуникаторов и КПК предназначен CDC, как более функциональный; для мобильников – CLDC.

Поскольку сотовые телефоны отличаются по устройству от компьютеров, потребовался API, обеспечивающий работу GUI и хранение настройки приложений и поддержку других специфические для мобильников возможности. Эту задачу берет на себя API под названием Mobile Information Device Profile (MIDP). На данный момент существует несколько версий: MIDP 1.0 был создан еще в 2000 году и накладывал ряд серьезных ограничений, поэтому в 2002 была выпущена новая версия, MIDP 2.0 использующаяся до сих пор.

Необходимо отметить, что список встраиваемых в телефоны API не ограничивается CLDC и MIDP. Существует целый ряд спецификаций мобильных стандартов Java, разработанных по запросам на спецификацию (JSR – Java Specification Request), которые также реализованы в качестве API и обычно носят номер соответствующего им запроса.

Возможности Java ME

Возможности каждого отдельного телефона можно описать несколькими параметрами, самым главным из которых будут поддерживаемые API. Основные API – это уже описанные CLDC и MIDP. Но кроме них есть и множество других. Некоторые используются широко, некоторые – не очень.

Например, заявленная поддержка Java 3D для телефона означает поддержку JSR 184 – Mobile 3D Graphics API (сокращенно M3G), а за возможности работы java-мидлета с видео и аудио отвечает JSR 135 – Mobile Media API.

Преимущества и недостатки Java ME

Java обладает следующими достоинствами:

- Один из главных плюсов Java ME – полное отсутствие вирусов и червей и очень небольшие возможности для троянских программ. Если исключить случаи некорректной реализации Java-машин, существование вирусов и червей на Java ME принципиально невозможно. Дело в том, что для всех Java-машин на современных телефонах такие действия, как посылка sms или передача данных через Bluetooth должны подтверждаться пользователем, что практически блокирует любые возможности по работе на java-машине вредоносного кода.

- Другой плюс Java ME – гибкость реализации. Каждый производитель решает, какой именно функционал поддерживается тем или иным телефоном. С другой стороны, это создает определенные сложности для разработки мидлетов из-за большого разнообразия аппаратов.

Недостатки Java являются:

- Работа с устройством через API. В обычной программе, которая выполняется через API операционной системы, команды выполняются процессором, и тут контроль над действиями программы никто не осуществляет. Возможности Java сильно ограничены заложенными API и если аппарат не обладает нужной функциональностью, то сделать ничего нельзя.
- Другой минус кроется в CLDC, являющейся сокращенной версией API Java SE. Этот API был разработан как набор базовых функций для телефонов, и поэтому функциональность API была серьезно урезана. В настоящее время, практически все телефоны поддерживают MIDP 2.0, заметно более требовательный к аппаратному обеспечению, однако это не привело к появлению новых версии CLDC.
- С этим связан и еще один недостаток: в отличии от Java SE, где постоянно совершенствуются сами возможности языка, добавляются новые средства, мобильная Java не изменяется со времени создания версии 3.

1.4.4. Операционная система iPhone OS

Такие мобильные устройства, как коммуникатор iPhone и плейер iPod Touch, работают под управлением операционной системы iPhone OS, являющейся производной от «настольной» Mac OS X.

Пользовательский интерфейс

Интерфейс iPhone OS основан на принципе управления устройством с помощью пальцев с использованием специальных жестов. Элементами пользовательского интерфейса являются слайдеры, переключатели и кнопки. Операционная система понимает такие жесты, как прикосновение, множественные прикосновения, жесты увеличения и уменьшения. Кроме того, использование встроенного акселерометра позволяет автоматически менять ориентацию экрана с вертикальной на горизонтальную.

Рабочий стол (рис. 1.10) содержит список всех установленных на устройстве приложений и «док» в нижней части экрана, показывающий приложения, с которыми пользователь работает чаще всего, который появляется при включении устройства или нажатии единственной аппаратной кнопки. На экране также присутствует строка состояния, показывающая время, уровень заряда батареи и силу сигнала.

Рис. 1.10. Рабочий стол iPhone

В iPhone не существует единой концепции запуска или завершения текущего приложения, кроме запуска приложения с рабочего стола и выхода из приложения для возвращения к рабочему столу. Хотя определенная многозадачность существует, функциональность ее ограничена: родные приложения могут работать в фоновом режиме, сторонние закрываются при выходе. Многие из встроенных приложений созданы для совместной работы; что позволяет передавать данные из одного в другое (так телефонный номер может быть извлечен из почтового сообщения и его можно сохранить в качестве контакта).

Поддержка приложений

Web приложения

Браузер Safari, встроенный в iPhone OS, с самого начала поддерживал работу JavaScript, в т. ч. и приложений, написанных с применением AJAX. Поэтому первоначально предполагалось, что iPhone будет работать исключительно с подобными браузерными «программами», о чем и было объявлено в 2007 году на WWDC – основной конференции компании Apple.

Набор средств разработки

В марте 2008 года Apple выпустила полноценный SDK, позволяющий разрабатывать программы для iPhone и iPod Touch и тестировать их на встроенном эмуляторе. Однако, эмулятор доступен только для тех, кто обладает платным членством в Apple Developer Connection.

Распространением программ занимается Apple: легально установить на устройства можно только приложения, приобретенные на сайте App Store. Разработчики могут устанавливать любые цены на свое ПО, из которого Apple получает 30 % от стоимости, или распространять их через сайт бесплатно (не считая оплаты членства в ADC).

Состав средств разработки SDK

В связи с тем, что iPhone основан на том же ядре Mach, что и Mac OS X, набор инструментов для разработки ПО для iPhone основан на XCode. SDK состоит из следующих частей:

- Cocoa Touch, отвечающий за обработку прикосновений, жестов и элементов управления, использование акселерометра, локализацию и использование встроенной камеры;
- Media отвечает за OpenAL,работу с фото, аудио и видео, Quartz, Core Animation, OpenGLES
- Core Services работает с сетью, встроенной БД SQLite, отвечает за пространственное позиционирование;
- OS X Kernel выполняет ключевые функции ОС: TCP/IP, сокеты, управление питанием, работа с файолой системой и обеспечение безопасности.

Необходимо также отметить, что работает SDK исключительно под управлением Mac OS X.

Язык Java на iPhone

Компания Sun Microsystems анонсировала планы по созданию виртуальной машины Java на iPhone, на основе J2ME. Однако на данный момент нет никаких данных о поддержке этого проекта со стороны Apple, что, учитывая особенности модели распространения программного обеспечения для данной платформы, не позволяет говорить о возможностях запуска Java на iPhone.

1.4.5. Операционная система Palm OS

Palm OS – операционная система для наладонных компьютеров и коммуникаторов. Под управлением ОС Palm OS работают около 39 млн устройств, произведенных Palm Inc. и другими производителями.

Интересная особенность Palm OS то, что ядро ОС, лицензированное у компании Kodak, многозадачное, а для пользователя ОС однозадачная, хотя и с возможностью фонового проигрывания музыки, MP3 и т. п. (проще говоря – одновременно на экране может отображаться лишь одно приложение). Более того, условия лицензионного соглашения, запрещают Palm раскрывать сторонним разработчикам API для создания фоновых задач на уровне ядра.

Наиболее популярная версия – Palm OS Garnet (5.4.x), на этой версии строятся все современные КПК Palm.

Рис. 1.11. Основной экран PalmOS

Разработка приложений

Основным языком для разработки программ для Palm OS Garnet является C/C++. Для них официально поддерживается два

компилятора: коммерческий продукт, CodeWarrior Development Studio для Palm OS и набор инструментов с исходным кодом под названием prc-tools, основанный на старой версии GNU Compiler Collection (gcc). Недостатками CodeWarrior является его высокая стоимость и то, что дальнейшее развитие этого проекта прекращено, в prc-tools же отсутствует несколько функций коммерческого компилятора. Версия prc-tools включена в бесплатный набор разработчика Palm OS Developer Suite (PODS).

Существуют инструменты разработки, которые не требуют низкоуровневого программирования на C/C++, такие как PocketC/PocketC Architect, CASL, AppForge Crossfire (который использует Visual Basic, Visual Basic.NET, или C#), Handheld Basic, Pendragon Forms и Satellite Forms (с похожим на Visual Basic синтаксисом). До 2008 года для PalmOS существовала виртуальная машина Java, теперь ее поддержка прекращена. Для PalmOS существует, однако, java-подобная виртуальная машина SuperWaba. Две среды для программирования на паскале под Palm OS. Некоммерческий PP Compiler работает прямо на устройстве, а PocketStudio является Delphi-подобным IDE для настольных компьютеров под управлением Windows, которая имеет свой конструктор форм и генерирует prc файлы.

1.4.6. Symbian OS и платформа Series 60

Самой распространенной операционной системой для смартфонов в настоящее время является Symbian OS. Согласно данным Canalys к первому кварталу 2008 года Symbian OS занимала 59 % рынка умных мобильных устройств – это 226 миллионов проданных аппаратов по всему миру. Ключевой причиной подобного успеха является то, что именно эта ОС используется крупнейшим производителем смартфонов – компанией Nokia.

Основной программной платформой для смартфонов является S60 Platform. Она была разработана компанией Nokia на основе операционной системы Symbian и лицензирована ей другим производителям, таким как Lenovo, LG Electronics, Panasonic и Samsung.

Symbian – это компания по лицензированию программного обеспечения, которая разрабатывает и поставляет передовую, открытую, стандартную операционную систему – ОС Symbian – для мобильных телефонов (смартфонов).

Symbian была основана как независимая частная компания в июне 1998 г., задаче которой является установлении ОС Symbian в качестве мирового стандарта для мобильных цифровых систем обмена данными, и в первую очередь для использования в сетях сотовой связи.

S60 состоит из набора библиотек и стандартных приложений, таких как телефония, электронная записная книжка и плейер.

Программное обеспечение для S60

Платформа S60 позволяет запускать приложения, разработанные под J2ME, C++ и Python. При разработке приложений необходимо учитывать, что релиза S60 было три: «Series 60» (2001), «Series 60 Second Edition» (2004) и «Series 60 3rd Edition» (2005).

В первом издании S60 разрешение экрана было фиксированным: 176x208. Со Feature Pack 3 второго издания S60 начало поддерживать различные разрешения, например Базовое (176×208), QVGA (240×320) и Двойное (352×416).

Необходимо отметить, что программное обеспечение, написанное для первой и второй версии S60 бинарно несовместимо с третьей версией S60 из-за того, что там используется новая версия Symbian OS (v9.1).

Структура OS Symbian

Структура Symbian OS подобна большинству настольных операционных систем с такими функциями, как приоритетная многозадачность, многопоточной обработкой и защитой памяти. Преимуществом системы является нацеленность ее функций на переносные устройства с ограниченными ресурсами, которые должны работать в течении многих месяцев и лет. К таким функциям относится минимизация использования памяти и редкость ее утечки. Так же следует отметить оптимизацию использования CPU в целях снижения энергопотребления – процессор отключается, когда приложения находятся в режиме ожидания. Структуру же самой системы можно условно разбить на уровни.

На самом низком уровне находятся основные компоненты Symbian, состоящие из ядра (EKA1, для Symbian 6.0-8.1a, или EKA2 для Symbian 8.1b-9.5) и пользовательская библиотека, позволяющая программам сторонних производителей обращаться к функциям ядра. Symbian ОС имеет микроядерную архитектуру, что означает, что ядро содержит минимум необходимых функций операционной системы. К этим функциям относится поддержка многозадачности и управление памятью.

Поддержку файловой системы и сетевой составляющей осуществляют серверы пользовательской стороны, расположенные на базовом уровне. Файловый сервер обеспечивает подобное DOS отображение файловой системы устройства (каждому диску назначена определенная буква, обратный слеш определяет каталог). Symbian поддерживает различные типы файловых систем, включая FAT32 и специфическую для Symbian систему NOR.

На уровень выше файловой системы находятся системные библиотеки, которые выполняют такие функции как преобразование кодировок символов, поддержку базы данных DBMS и обработку файлов ресурсов. Остальное программное обеспечение находится на верхнем уровне.

Средства разработки для Symbian OS

Основным языком программирования для Symbian OS является C++ (даже сама система написана на этом языке). Вместе с тем, многие устройства на Symbian OS могут быть программируемы на OPL, Python, Visual Basic, Simkin и Perl, а также Java ME и PersonalJava.

Фирмы-разработчки устройств на Symbian как правило предоставляют комплект средств разработки (SDK) для своих устройств. Естественно, что для каждой платформы они различны. Подробнее о доступных языка программирования и SDK для платформы Series 60 смотрите в разделе, посвященном Series 60.

Вопросы для самопроверки

1. Дайте определение мобильных вычислительных устройств?
2. В чем преимущества использования мобильных устройств?
3. Какие типы мобильных устройств существуют?
4. Какие операционные системы используются в мобильных устройствах?
5. Какие типы памяти применяются в мобильных устройствах? В чем заключаются их достоинства и недостатки?

2. МОБИЛЬНЫЙ БИЗНЕС И СЕТЕВЫЕ СЕРВИСЫ

Ориентированные на сервисы архитектуры построения информационных систем основываются на использовании сетевых сервисов. Под сетевым сервисом понимается принцип организации информационно программной системы, в которой выделяют поставщиков услуг и потребителей услуг. Поставщиками услуг обычно является стационарные компьютеры-серверы, на которых выполняется программное обеспечение с полезной для потребителей функциональностью, контент (базы данных, документов, изображений и пр.), и поддерживающим открытые протоколы доступа (например, такие как HTTP и WSDL) для взаимодействия с пользователями.

В мобильных сервисных системах потребители услуг являются пользователи, применяющие мобильные вычислительные устройства для подключения по открытым стандартным протоколам к сетевым сервисам, предоставляемым провайдерами.

2.1. Определение и возможности мобильных технологий

Мобильный бизнес является логическим продолжением электронного и часто понимается как его подмножество. Обычно под мобильным бизнесом понимаются традиционные услуги электронного бизнеса, только предоставляемые через беспроводные сети. Однако мобильный бизнес обеспечивает и принципиально новые сервисы, которые не могут быть реализованы с помощью обычной сети. Например, локализация (определение местоположения), персонализация и оказание услуг в произвольном месте не могут осуществляться с помощью проводного Интернет-соединения. Таким образом, мобильный бизнес не является подмножеством электронного бизнеса, а выходит за его границы, предоставляя абсолютно новые возможности и создавая при этом совершенно новые проблемы (рис. 2.1).

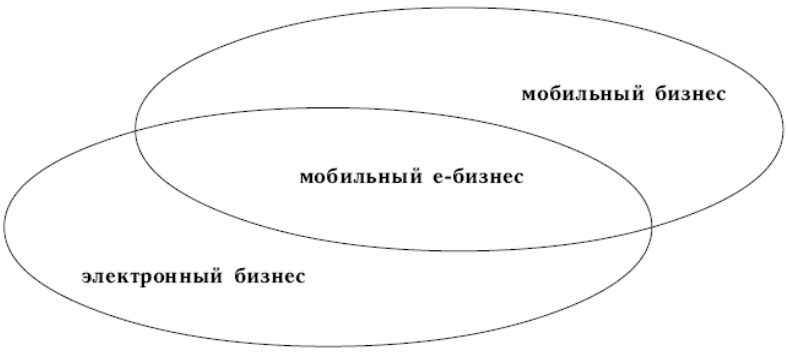

Рис. 2.1. Электронная и мобильная коммерция

Обычно область пересечения мобильного и электронного бизнеса называют мобильным е-бизнесом, подразумевая при этом такие мобильные технологии, которые реализуются на базе традиционных сетевых структур. Исходя из исследований в этой области, под мобильным бизнесом будем понимать обмен товарами, услугами и информацией, который осуществляется с использованием мобильных устройств.

2.2. Особенности мобильного бизнеса

Главной отличительной чертой мобильных технологий является возможность связаться с пользователями независимо от того, где они находятся, а для самих пользователей – доступность услуг в любое время из любой точки земного шара. Поэтому основными преимуществами мобильных услуг обычно называют следующие четыре драйвера: мобильность, достижимость, локализация и идентификация.

Мобильность – одна из основных особенностей рассматриваемых технологий. Беспроводные устройства позволяют пользователям получить доступ к необходимой информации практически с любого места. Поскольку мобильные устройства постоянно находятся при своих владельцах, то последние могут посылать и получать информацию независимо от времени и своего местопребывания. Причем к преимуществам беспроводных технологий относится не только предоставление доступа их владельцам к любой необходимой информации. Эти технологии также дают возможность связаться с их пользователями и другим людям. Такая услуга является особенно важной для руководителей, которым требуется постоянная связь со своими сотрудниками, находящимися вне офиса, для того чтобы сообщать им новые задачи и инструкции. Точно так же многим обычным пользователям важно всегда иметь связь с членами своих семей и друзьями.

Благодаря инфраструктуре локализации местонахождение любого человека, имеющего при себе мобильное устройство, может быть легко установлено. Такая способность определения местоположения пользователя может оказаться очень важной, например, для служб, предоставляющих информацию о транспортных пробках, ближайших местах парковки, отелях или ресторанах. Многие компании могут быть заинтересованы в том, чтобы иметь возможность определить, где находится их служебный транспорт или, например, ближайший к месту аварии клиента специалист. Кроме возможности локализации пользователя мобильного устройства также возможна и его идентификация. Так как обычно между мобильным устройством и человеком существует взаимосвязь типа один к одному, то беспроводной терминал может авторизовать пользователя для доступа на охраняемую территорию или для совершения операции оплаты.

В будущем вполне возможно появление приложений, использующих идентификацию пользователя и предоставляющих функции «мобильного кошелька» или мобильного ключа от офиса или дома

2.3. Мобильные приложения

Успех всякого приложения определяется не только самими по себе новыми технологиями, но и способностью их правильного применения. Пользователям интересны не просто технологии, их привлекают новые способы решения существующих проблем. Так что беспроводные приложения будут привлекательны только в том случае, если они полезны конечным пользователям.

Многие важные приложения мобильного бизнеса возникли в результате интеграции беспроводных технологий с уже существующими электронными бизнес приложениями. Одновременно уникальные свойства мобильного бизнеса – мобильность, локализация, идентификация и т. д. – дают толчок для развития новых приложений и бизнес-моделей.

2.3.1. Виды мобильных приложений

Мобильные приложения можно подразделить на три группы: приложения для поддержки мобильных технологий (enabling applications), приложения для потребителей (consumer applications) и бизнес-приложения (business applications). Приложения для поддержки мобильных технологий в основном представляют собой модификации программ, используемых в электронном бизнесе для управления информацией и организационной деятельности. В числе прочих эта группа приложений включает в себя мобильную почту, мгновенную передачу сообщений, программы-ежедневники и мобильный офис. Приложения для поддержки мобильных технологий также называются горизонтальными приложениями, так как они могут использоваться как обычными, так и бизнес-пользователями. Причем, если частные лица используют их, прежде всего, для получения последних новостей и для того, чтобы поддерживать связь со своими семьями и друзьями, то управляющим и мобильным служащим (Мобильный служащий – работник компании, выполняющий свои служебные функции вне офиса) они нужны прежде всего для получения доступа к внутренним справочникам компании, системам передачи сообщений, к спискам контактов и т. д.

Следующая группа – это приложения для потребителей, представляющие собой достаточно большую и важную группу мобильных приложений. Во-первых, к этой группе относятся сервисы, предлагаемые традиционным бизнесом и стремящиеся расширить свое присутствие и на мобильную сферу. Различные финансовые услуги, как, например,

выполнение банковых операций и платежей, услуги охраны, мобильные покупки, реклама, развлекательные услуги, равно как и услуги по предоставлению мобильных новостей и информации – лишь малая часть из длинного перечня подобных мобильных решений.

Рассмотрим подробнее третью группу приложений, которая особенно важна для предприятий и организаций, т. е. обратимся к **мобильным бизнес-приложениям**.

Бизнес-решения можно разделить на приложения для покупателей, бизнес партнеров и служащих. Приложения для служащих, в свою очередь, могут быть подразделены на несколько категорий, таких как продажи и автоматизация управления сотрудниками, мобильное управление командировками и т. п. Многие компании в первую очередь внедряют программы для мобильных служащих, так как они обещают быстрый рост производительности и снижение затрат. Более того, такие приложения удобны для проведения экспериментов, мониторинга проектов и накопления необходимого опыта.

Мобильные бизнес-приложения довольно специфичны для процессов, лежащих в основе деятельности компании, и поэтому их часто называют вертикальными. При создании мобильных бизнес-решений разработчикам необходимо учитывать особенности бизнес-процессов каждой компании. Даже если процессы очень похожи, способы их реализации в разных компаниях могут сильно отличаться. Это означает, что мобильные приложения достаточно сложно стандартизировать. В любом случае их приходится разрабатывать с нуля, тщательно описывая схему бизнес-процессов и их взаимосвязи в компании. Напротив, приложения для поддержки мобильных технологий и приложения для потребителей стандартизируются не так сложно и в будущем могут стать продуктом массового потребления.

Мобильное управление взаимоотношениями с клиентами

Приложения по управлению взаимоотношениями с клиентами (Customer relationship management – CRM) предназначены, прежде всего, для улучшения качества тех функций компаний, которые связаны с потребителями, например, маркетинг, функции продажи и предоставления сервисов. Мобильные CRM приложения повышают качество обслуживания клиентов, что ведет к повышению доверия к продукту и увеличению удовлетворенности покупателей. Традиционные CRM решения здесь не подходят из-за существующего разделения функций работников, находящихся в офисе, и работников, взаимодействующих с клиентами вне его. Поэтому информация не может быть полностью и быстро получена в исходной базе данных, точно так же как она не

может быть полностью представлена и обработана в той точке, где оказывается услуга. До настоящего времени торговым представителям и техническим специалистам часто приходится сначала объединять полученную информацию на бумаге, а уже потом вводить ее в ПК. В результате часть информации забывается, теряется, а оставшаяся часть может содержать ошибки. Это ведет к появлению в базах данных неточной и противоречивой информации.

Кроме того, важные сведения о потребителях, содержащиеся в центральной базе данных, не могут быть эффективно использованы работниками, находящимися вдали от офиса, так как им доступна только часть информации. Однако высокая конкуренция требует своевременного появления достоверной и полной информации у всех работников для того, чтобы пожелания клиентов выполнялись точнее, быстрее и лучше.

Для реализации этих задач используются мобильные CRM решения, которые позволяют собирать информацию обо всех контактах с клиентами и объединять ее в общий банк данных. Эти данные применяются при создании профиля клиента и проведении целевых маркетинговых акций. Торговые представители, владеющие CRM приложениями, имеют возможность получать информацию о предыдущих контактах с клиентом, использовать архивы, информацию о продажах, ценах, доставке. Точно так же работники, находящиеся вне офиса, могут получать на свои мобильные устройства интересующие их технические детали, другую информацию. Все эти данные помогают повысить эффективность продаж, качество работ, осуществляемых вне офиса, и, что самое главное, – улучшить взаимодействие с клиентами.

Мобильное управление цепочками поставок.

Цель стратегии управления цепочками поставок (Supply chain management – SCM) – обеспечить непрерывность финансовых, информационных и материальных потоков между поставщиками, бизнес-партнерами и потребителями.

Мобильные технологии могут существенно улучшить обмен информационными потоками между фронт- и бэк-офисами компании за счет усовершенствования механизмов сбора данных и механизмов оповещения заинтересованных лиц о наступлении некоторых событий. Например, беспроводные сканеры штрих-кодов или технологии распознавания частот радиоволн могут быть использованы для сбора данных в точке появления информации. Вместе с тем собранные данные должны быть переданы в центральный банк хранения данных и в режиме реального времени распространены во все заинтересованные подразделения компании. Это помогает компании принимать более обоснованные и компетентные реше-

ния, а также улучшать качество планирования и прогнозирования. Другие возможности SCM приложений позволяют управлять материально-техническим снабжением, запасами, внутрипроизводственными и потребительскими заказами, лучше определять направления процессов, быстрее передавать распоряжения, а также предоставляют возможность распространять приказы, уведомления, доклады максимально удобным способом. Внедрение приложений мобильного бизнеса может пригодиться там, где приходится координировать работу большого количества людей. Такие перекрестно-организационные решения могут быть использованы, например, в больницах или аэропортах.

Мобильная автоматизация продаж

Сегодня торговые представители обеспечивают компаниям прибыль в значительной степени благодаря мобильным технологиям, предоставляющим им полные сведения о покупателях и товарах. Раньше торговому агенту приходилось сначала записывать пожелания клиента на бумаге, а затем обсуждать их с офисными служащими для уточнения заказа. Далее торговый представитель готовил итоговое предложение и посылал его клиенту. После получения подтверждения торговый представитель отправлял заказ в отдел продаж. При использовании мобильных технологий формирование заказа осуществляется гораздо быстрее, поскольку существует возможность обсудить детали с офисными служащими у покупателя дома. Сведения о количестве запасов, датах и условиях поставок, последних ценах и специальных акциях, получаемые прямо в присутствии клиента, значительно ускоряют процесс заключения сделки, а также сокращают количество претензий от клиентов. Причем помимо получения информации от своей собственной компании, торговый агент также может оценивать цены и качество товаров конкурентов. В результате торговым представителям легче реагировать на любые изменения на рынке и предлагать клиенту максимально выгодные условия.

Мобильная автоматизация управления совместной работы территориально удаленных подразделений

Работники компании, обслуживающие клиента вне офиса, – наиболее мобильная группа служащих компании. Хорошее обслуживание в месте продажи предполагает быстрое реагирование на поступающие заказы, гибкую систему предложений, профессиональное устранение возможных ошибок. Для работников, находящихся вне компании, очень важны беспроводной доступ к важной информации и возможность взаимодействия со специалистами, находящимися в головном офисе. Более того, работа на выезде всегда сопровождается большим количеством

бумажных источников информации, что сильно понижает качество данных и зачастую связано с возникновением ошибок, приводит к лишним затратам. Бумажная волокита значительно замедляет весь процесс сделки, делая его непрозрачным и сложным.

Можно выделить три основных преимущества для служащих, работающих с клиентами вне офиса. Во-первых, они получают прямо на свои мобильные устройства, текущие поручения со всеми соответствующими деталями, включая необходимые контактные данные. В результате исчезает необходимость в постоянных поездках в офис и обратно. Во-вторых, будучи на связи, служащие имеют постоянный доступ к информации по предыдущим эксплуатационным и восстановительным работам, к архиву услуг, а также к сведениям о товарах и деталях. Более того, можно постоянно иметь «под рукой» огромные каталоги запасных частей, в которых не сложно найти информацию, необходимую для клиента. Технические специалисты получают мгновенный доступ к всевозможным инструкциям, сопроводительным документам и пошаговым руководствам.

И, наконец, в-третьих, беспроводные технологии предоставляют служащим очень эффективные и комфортабельные условия для ведения отчетности по заключению сделок, затратам времени и материальным затратам. Подобные сервисы включают такие услуги, как проверка наличия запасных деталей и их классификация, заполнение форм осмотра товаров и формирования счетов в режиме реального времени (on-line).

Телеметрия и мобильный удаленный контроль.

Под «телеметрией» обычно понимают «измерение и автоматическую передачу данных из отдаленных источников по проводам, по радио или с помощью других средств» (WordNet). Телеметрия относится к приложениям типа B2M (бизнес – машина), так как фактически передача информации происходит без вмешательства человека. Благодаря мобильным технологиям контроль оборудования и механизмов можно осуществлять на расстоянии. Такие технологии преимущественно используются там, где техника недоступна для традиционных проводных сетей, например, на гидроэлектростанциях, нефтяных месторождениях или в других отдаленных местах. Еще одно место применения удаленного мобильного контроля – копировальные или торговые автоматы, которые могут автономно докладывать о необходимости пополнения расходных материалов или о поломке. Обычно такие автоматы сильно территориально разбросаны и на их проверку требуется много времени и средств. Удаленный мобильный контроль может использоваться в зданиях и машинах для обеспечения их безопасности. В случае выяв-

ления проблемы сенсоры просто высылают владельцу сообщение с указанием места и вида проблемы.

Вспомогательные приложения.

К вспомогательным бизнес-приложениям относится целый ряд мобильных решений, которые облегчают работу за счет своевременной передачи необходимой информации в нужное место и в нужной форме. Такие приложения позволяют лицам, принимающим решения, оперативно реагировать на эту информацию. Среди прочих эта группа включает в себя мобильное управление поездками (командировками) и интеллектуальные бизнес-приложения. Служащие, часто выезжающие в командировки, тратят огромное количество времени и средств на организацию своих поездок, им приходится ориентироваться в постоянно изменяющихся часовых поясах, населенных пунктах и расписаниях. Мобильные приложения могут упростить им работу, предоставив больше времени для основной работы.

Кроме того, мобильные интеллектуальные приложения предоставляют оперативные данные ответственным лицам. С их помощью менеджеры получают возможность отслеживать ключевые показатели деятельности и текущие бизнес-операции, а также использовать сигналы тревоги, предупреждения и возможности составления отчетов.

2.3.2. Стоимость мобильных приложений

Электронные и традиционные бизнес-приложения замещаются мобильными бизнес-приложениями, только если они приносят бизнесу дополнительную прибыль. Недостаточно просто адаптировать существующие электронные приложения к мобильным устройствам. Новые мобильные бизнес-платформы выживут, только если они будут предлагать достаточные преимущества. Чтобы оценить достоинства мобильных приложений, необходимо определить, какие возможности, не реализуемые электронными и традиционными бизнес-приложениями, они предоставляют. Дополнительная ценность мобильных приложений вызвана возможностью своевременно предоставлять пользователям оперативную информацию и возможностью вести дела из любого места в любое время. Привлекательность беспроводных методов решений зависит от двух факторов:

- во-первых, насколько управленческие решения в компании зависят от оперативности данных;
- во-вторых, насколько компании важна мобильность служащих.

Чем более существенны эти два фактора, тем выше ценность мобильных приложений. Следовательно, новизна мобильных технологий

заключается в их способности предоставить своевременную и важную информацию. Стоимость информации определяется как разность между доходами, полученными при принятии решения в отсутствии информации, и доходами, которые могут быть получены при наличии информации. Делая доступными необходимые данные для лиц, принимающих решения, мобильные приложения «сокращают дорогостоящие и длительные задержки в традиционных бизнес-процессах». Например, обновленная информация о спросе сразу попадает к торговым представителям, находящимся вне офиса, что значительно улучшает качество цепочки поставок и существенно снижает затраты.

2.4.1. Прибыльность мобильных решений

Применение мобильных технологий на практике должно восприниматься как обычный IT-проект. Перед внедрением мобильных решений на практике должны быть подсчитаны и оценены все связанные с ними затраты и прибыли. Компаниям не следует срочно переносить свои бизнес-операции на мобильную основу только потому, что обновление «обещает» значительное снижение затрат и высокий рост производительности. На самом деле достаточно трудно предложить количественные доказательства того, что мобильные методы улучшат работу компании. Каждая компания должна взвесить и тщательно проанализировать все те возможности, которые беспроводные приложения могут им предоставить. Только в этом случае введение мобильных методов может оказаться действительно прибыльным и способным привести к реальным улучшениям бизнес-операций. Чтобы оценить итоговую прибыльность проекта, компании необходимо сопоставить соответствующие доходы и затраты.

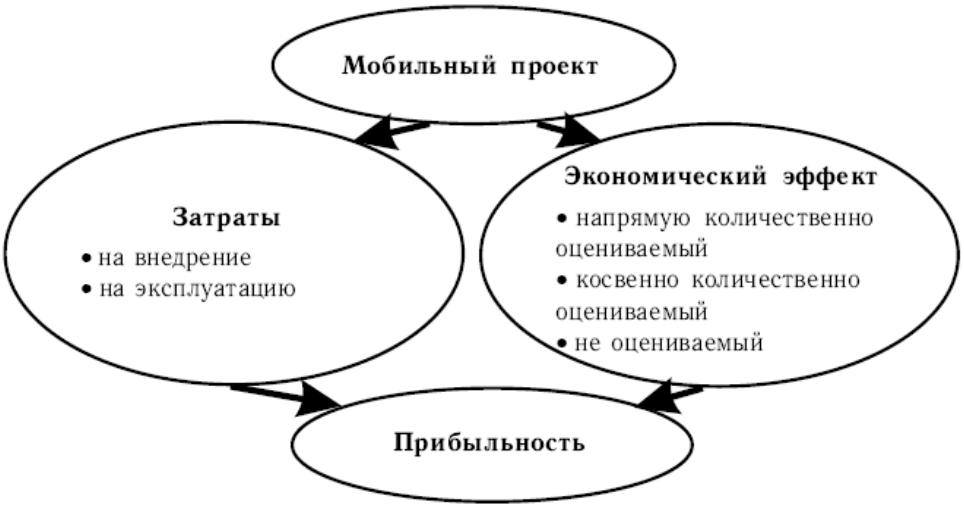

Рис. 2.2. Оценка прибыльности мобильных проектов

Но следует иметь в виду две проблемы. Во-первых, у многих компаний есть тенденция недооценивать затраты, связанные с реализацией и интеграцией проекта. Во-вторых, очень трудно количественно определить выгоды мобильных решений и представить их в денежном эквиваленте. Повышение качества работы с клиентами, улучшение качества всей последовательности выполняемых работ, так же как и улучшение имиджа компании, могут повлиять на доход только косвенно. Однако они могут существенно увеличить конкурентные преимущества компании, а это тоже не следует сбрасывать со счетов. Взаимосвязь между затратами и доходами показана на рис. 2.2.

2.4.2. Затраты на выполнение мобильных проектов

Обычно, говоря о затратах, связанных с мобильными приложениями, на первое место ставят инвестиции в мобильные устройства и оборудование. Однако, как показывают исследования [???], эти затраты составляют менее трети всех расходов проекта. Более того, стоимость оценки проекта, его реализации и интеграции в существующую IT-инфраструктуру составляет наиболее дорогую часть мобильных решений. Поэтому, выбирая разработчика для реализации мобильного проекта, организации не следует сосредоточиваться только на стоимости технических компонентов. Наиболее важным критерием выбора должны стать компетентность и опыт поставщика услуг.

Все затраты на мобильные проекты могут быть разделены на две группы – расходы на запуск системы и эксплуатационные расходы (табл. 2.1).

В начале проекта имеют место расходы на его запуск. Именно они включают оценку проекта, затраты на планирование и внедрение. К этой же группе относится и закупка необходимого мобильного оборудования, посреднического программного обеспечения, специальных мобильных приложений. На следующем шаге компании необходимо сгруппировать все новое оборудование и приложения, а затем интегрировать их в существующую IT-систему. Эти затраты во многом зависят от существующей IT-инфраструктуры и структуры бизнес-процессов в самой компании. И, наконец, компании следует вкладывать деньги в разъяснение целей, мотивацию и образование своих сотрудников.

Вторая группа издержек состоит из эксплуатационных затрат, которые будут произведены в течение всего жизненного цикла проекта. Они, в свою очередь, делятся на три подгруппы.

Во-первых, это затраты на системное администрирование и управление, включающие в себя реконфигурацию и поддержку системы,

а также помощь служащим, потребность в которой существует постоянно. Во-вторых, эксплуатационные расходы включают в себя расширение новой мобильной платформы, интеграцию дополнительных приложений и привлечение новых служащих. Поддержка, ремонт и закупка запасных частей – все это тоже входит в рассматриваемую подгруппу. Наконец, последняя подгруппа включает расходы на оплату мобильной связи и передачу данных по беспроводным телекоммуникационным сетям.

Таблица 2.1

Затраты на мобильный проект

	Приобретение	Интеграция и реализация	Упорядочение бизнес процессов
Расходы на разработку и внедрение системы	• мобильные устройства; • другое оборудование; • программное обеспечение • мобильные приложения.	• оценка проекта, планирование и управление; • реализация; • интеграция в существующую IT-систему.	• обновление бизнес процессов; • мотивация и обучение служащих.
Текущие расходы	Администрирование и управление системой	Обновление и развитие	Передача данных
	• хостинг; • управление правами доступа; • поддержка служащих; • конфигурирование оборудования.	• вовлечение новых служащих; • интеграция дополнительных приложений; • затраты на покупку запасных частей.	• оплата услуг по передачи данных; • оплата соединений.

2.4.3. Преимущества мобильных сетевых сервисов

Несмотря на затраты, беспроводные приложения обеспечивают большие преимущества. Причем существует три направления, где наиболее привлекательно использование мобильных решений.

Во-первых, беспроводные технологии помогают связаться тем работникам, которые проводят основную часть своего рабочего времени вне компании, с центральной IT-системой. В настоящий момент для получения нужной информации и при необходимости передать отчет о сделке мобильные служащие используют Интернет, соединяясь с центральным сервером с помощью модемов, посылая сообщения, факсы, и т. д. Конечно, такие служащие будут активно использовать улучшенные возможно-

сти мобильных технологий для связи с компанией через беспроводные сети в любое время с любого места. А если у компании уже есть положительный опыт использования удаленного доступа, то они будут гораздо легче и охотнее переключаться на появляющиеся технологии.

Во-вторых, мобильные решения могут оптимизировать последовательность выполняемых в компании технологических действий. Они позволяют автоматизировать некоторые шаги в существующих процессах, постепенно улучшая и оптимизируя их. Например, мобильные предупреждения могут использоваться для обнаружения препятствий, аварийных ситуаций, ускорения получения подтверждений. Сбор данных в едином хранилище данных с помощью мобильных технологий может привести к уменьшению необходимого уровня запасов товаров и снижения затрат. Правда, такие беспроводные приложения радикально не изменят способа ведения бизнеса.

И, наконец, беспроводные технологии, без сомнения, облегчат развитие совершенно новых бизнес-моделей, которые основаны исключительно на мобильных платформах. Эти модели будут эффективнее, чем те, которые используются в настоящее время, и смогут решать задачи совершенно новыми способами. К сожалению, сегодня слишком рано говорить о реальных мобильных бизнес-процессах. Жизнеспособные беспроводные решения сначала должны быть созданы и проверены временем и высокой конкуренцией, прежде чем их можно будет назвать успешными.

Широкий ряд преимуществ мобильных приложений нелегко оценить количественно. И если увеличение прибыли компании может легко быть выражено в денежном эквиваленте, то доход от совместного использования информации сотрудниками оценить очень трудно. Даже доходы от улучшения имиджа компании трудно оценить в денежном выражении. Подводя итог многих исследований, можно сделать вывод о наличии шести групп процессов и людей, которые наиболее выигрывают от использования мобильных решений: клиенты компаний, бизнес партнеры и служащие, бизнес-процессы и информационные потоки, и сама компания (табл. 2.2).

Хорошо известно, что любая компания наибольшее внимание уделяет своим клиентам. И практически все, что делает компания, делается для удовлетворения их интересов. Мобильные технологии предоставляют громадные возможности для дальнейшего улучшения взаимоотношений между покупателями и компанией.

Торговые представители, которые посещают заказчиков на дому или в офисе, могут более оперативно реагировать на пожелания и потребности клиентов. Используя мобильные устройства, соединенные с центральной базой данных, они могут сообщать покупателю об усло-

виях и сроках доставки товара, о специальных предложениях и других подобных вещах. Теперь агенты гораздо лучше информированы о некоторых редких товарах и их конфигурации, они делают взаимодействие между компанией и клиентом более приятным и согласованным.

Таблица 2.2

Преимущества мобильных проектов

Клиенты	**Служащие**
• Рост удовлетворенности клиента • Близость к клиенту • Новые каналы связи • Повышение качества услуг • Исчерпывающий доступ к информации о клиенте • Быстрая реакция на заказы, запросы, жалобы и т. д. • Высокая лояльность потребителей	• Повышение мобильности • Независимость от местонахождения • Повышение производительности • Концентрация на ключевых проблемах • Дополнительное время на звонки и запросы • Улучшение сотрудничества между работниками, работающими вне офиса • Доступ к корпоративным новостям, контактам, календарям и т. д. • Управление персональной информацией • Улучшение связи между управляющими и персоналом • Рост удовлетворенности служащих • Хороший имидж служащих
Компания	**Интеграция и реализация**
• Рост конкурентных преимуществ • Оптимизация бизнес-процессов • Снижение затрат • Уменьшение количества ошибок, неправильных поставок и услуг • Быстрая реакция на пожелания клиентов • Рост гибкости и надежности • Способность предлагать новые продукты и услуги • Возможность вхождения на новые рынки • Хороший имидж компании	• Усовершенствование цепочки поставок • Краткое время ожидания ответа • Высокая точность информации • Доступ в реальном времени к сведениям партнеров • Высокий уровень доверия между клиентами • Улучшенные коммуникации • Активное сотрудничество • Рост удовлетворенности бизнес-партнеров

Окончание табл. 2.2

Информационные потоки	Бизнес-процессы
• Независимость от места сбора данных • Немедленный доступ к информации • Высокая надежность информации • Доступ в режиме реального времени • Отсутствие дублирования информации • Прозрачность и разделение информации	• Быстрые и совершенные технологические маршруты • Обоснованные решения и анализ • Рационализация технологических процессов • Более гибкое размещение персонала • Передача заданий от офисных служащих работникам, работающим вне офиса

Кроме того, торговые представители собирают все необходимые данные прямо в присутствии клиента и немедленно переправляют их в компанию. Таким образом, процесс выполнения заказа ускоряется, снижается количество ошибочных поставок, и, что самое главное, покупатель доволен. Благодаря мобильным приложениям, у покупателей появляется дополнительный канал для связи с компанией. В то же самое время улучшенное взаимодействие и сотрудничество между клиентами и фирмой ведет к усилению лояльности клиентов.

И, наконец, благодаря мобильным устройствам, находящимся под рукой, торговые представители теперь могут получать всестороннюю информацию о заказчиках. Компания имеет возможность создать профиль клиента и может персонифицировать предложения товаров и услуг.

Не только клиенты, но и бизнес-партнеры компаний выиграют от использования мобильных решений. Бесспорно, поставщики смогут увеличить эффективность работы, если получат постоянный доступ к данным о запасах товаров и запчастей в компании. Полная, более точная и своевременная информация позволит партнерам лучше планировать свои действия и тем самым повысить их эффективность. Кроме того, немедленные сообщения о проданных товарах или требуемых услугах сделают более качественными прогнозы спроса и снизят необходимые уровни запасов. Кстати, более короткое время ответа и развитые коммуникации ведут к налаживанию сотрудничества между всеми участниками. В то же время улучшение взаимодействия, несомненно, приведет к повышению уровня доверия и развитию отношений между фирмами.

Третья крупная группа, выигрывающая от использования беспроводных приложений, включает в себя служащих компаний. Движущей силой мобильного бизнеса является взаимосвязь предприятие – персонал (B2E – Business to employee), другими словами, мобильный бизнес

внутри компании. B2E решения стимулируют постоянные улучшения внутренних процессов и обещают прибыль в основном тем компаниям, в которых служащие часто работают вне офиса.

Прежде всего, это относится к продавцам и персоналу, обеспечивающим обслуживание клиентов на выезде, к менеджерам и служащим, много перемещающимся как внутри, так и снаружи офиса компании. Благодаря беспроводным технологиям служащим легче поддерживать связь друг с другом, а также с офисом. В результате они лучше координируют свои действия и лучше выполняют свою работу. Получая точные и своевременные сведения, служащие тратят меньше времени на поиск и подтверждение необходимой информации. Таким образом, они могут сосредоточиться на своих основных обязанностях и выиграть лишнее время для дополнительных контактов с клиентами и выполнения заданий. Более того, служащие становятся более независимыми от своего местопребывания, так как с помощью мобильных устройств они могут получать доступ к новостям компании, контактам, сообщениям, календарям событий и т. д. С помощью мобильных технологий служащие получают простой доступ к своим персональным информационным системам управления, включающим предупреждения и сигналы тревоги, листы заданий. Все вышесказанное способствует большей удовлетворенности рабочих, улучшению имиджа служащих, а также ощутимо повышает продуктивность деятельности компании в целом.

Введение мобильных решений может оказаться полезным и для бизнес-процессов внутри компании. Часто последовательность технологических операций задерживается из-за проблем со сбором данных в точке их происхождения. Бумажный поток документов, повторные вводы данных и человеческий фактор вызывают задержки и препятствуют непрерывному потоку информации. Компьютеры и Интернет повысили эффективность работы офисных служащих; мобильный бизнес способен повысить эффективность работы служащих, работающих вне офиса. Мобильные технологии делают возможным сбор данных в местах их возникновения. Как следствие, все участвующие в этом процессе стороны получают доступ к точной информации.

Доступность и разделение информации ускоряют цепочку технологических процессов. Постоянный доступ к высоконадежной и актуальной информации является основой обоснованных решений и глубокого анализа. Более того, некоторые задания могут пересылаться от офисных служащих к служащим, работающим вне офиса, что позволяет более эффективно распределять человеческие и материальные ресурсы. Подводя итог, можно утверждать, что все перечисленные факторы ведут к лучшей организации и рационализации бизнес-процессов компании.

Если клиенты компаний довольны, бизнес-партнеры надежны, служащие работают производительно и бизнес-процессы организованы более эффективно, тогда и сама компания будет получать прибыль. Способность оперативно реагировать на пожелания клиентов увеличивает доход компании. Сокращение числа ошибок и неправильного предоставления услуг, равно как и оптимизация бизнес-процессов, снижает затраты компании. И, наконец, увеличение скорости выполнения операций повышает гибкость и надежность всей компании. Следовательно, компания может выходить на новые рынки, предлагать новые товары и услуги и получать новых заказчиков. Все эти факторы позволяют увеличить конкурентные преимущества компании и улучшить ее имидж в обществе.

2.4. Пример проектирования мобильного сервиса

Рассмотрим мобильный сетевой сервис [11], который относится к области туризма. Она помогает туристам, посещающим незнакомый город и имеющим свободное время, ознакомиться с этим городом. Система советует туристам, как им провести свободное время, учитывая персональные предпочтения и общий контекст. В данном примере проектирования сервисной системы важными вопросами являются персонализация и определение местоположения пользователей.

2.4.1. Концепция мобильного сервиса

Представьте, что вы приехали утром в незнакомый город и только что вышли из гостиницы. Вечером вы должны встретиться со своим знакомым, а теперь у вас есть свободное время познакомиться с городом. Для этого можно воспользоваться мобильным устройством и набрать адрес мобильной сервисной службы «Персональный гид», чтобы она предложила вам интересную программу на полдень, в соответствии с вашими личными предпочтениями. Данный сервис может определить, где вы в настоящее время находитесь, и если вы ведете ваше расписание на мобильном устройстве, то сервис может определить, где вы собираетесь быть вечером и в какое время. В результате только нескольких нажатий кнопок, данный сервис предложит вам список возможных программ (туристических маршрутов) в течение полудня. Каждая программа включает набор интересных мероприятий, которые соответствуют вашим интересам. Вы выбираете программу, которая нравится вам больше всего. Затем мобильное устройство говорит вам куда пойти или какой общественный транспорт вам потребуется. В последнем случае оно укажет вам ближайшую остановку транспорта. Во время движения к остановке, данный

сервис расскажет вам некотор факты об интересных местах, рядом с которыми вы проходите.

При формировании возможных программ мероприятий сервис «Персональный Гид» учитывает несколько факторов. Во-первых, он знает, где вы находитесь. Просматривая ваше расписание, он узнает, где вы должны быть вечером. Конечно, если вы не имеете никаких запланированных встреч, то можно будет указать место и время окончания прогулки самому. Сервис будет планировать маршрут, который имеет начальную точку, ваше текущее положение, и ваше конечное место, где вы должны быть вечером. Конечно, он будет учитывать также и временные ограничения.

Как было отмечен выше, сервис знает ваши личные (персональные) интересы. Они используются не только для выбора интересных мест вдоль вашего маршрута, но также и для планирования маршрутов. Представьте, что когда вы сделали запрос к сервису на формирование маршрутов, было 11:00. Сервис знает, что вы обычно обедаете между 12:00 и 13:00. Он будет это учитывать и планировать остановку в кафе в обычное для вас время. Если вы предпочитаете fast food больше чем итальянскую кухню, то он будет автоматически искать закусочную вдоль вашего маршрута. Если есть достопримечательности, которые точно соответствуют вашим персональным предпочтениям, но расположенные далеко, то тогда у вас не будет времени на обед в кафе. Сервис также покажет этот маршрут, но поместит его ниже в ранжированном списке. Также могут учитываться и другие данные (параметры, особенности), например, прогноз погоды.

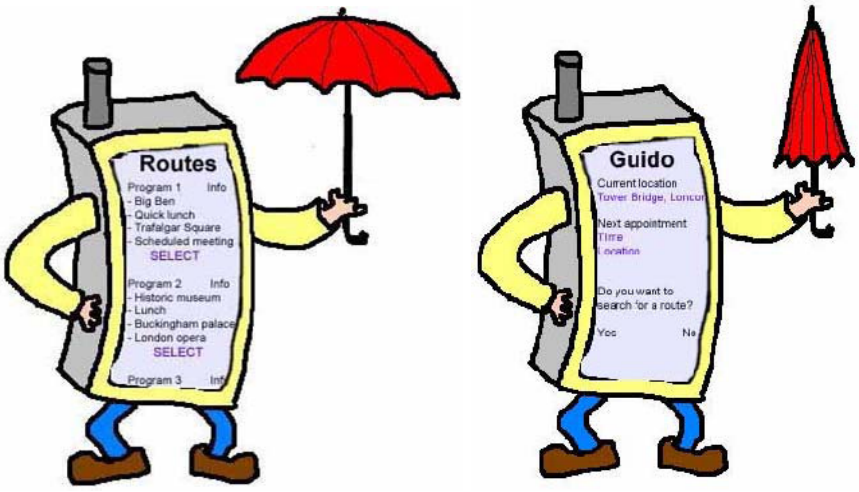

Рис. 2.3. Интерфейс сервиса «Персональный Гид»

Если сервисная система будет иметь информацию, о том, что вам нравится театр, и вы подтвердите, что хотите посетить его, то сервис забронирует для вас билет на подходящий спектакль.

Описание процесса использования сервиса

В результате нескольких щелчков по кнопкам КПК начинает работать мобильный сервис «Персональный Гид», который предлагает набор возможных маршрутов (рис. 2.3).

Пользователь выбирает из списка маршрут, который подходит ему больше всего ему. После этого он следует предлагаемому маршруту с помощью, предоставляемой на экране КПК карты (рис. 2.4). Когда пользователь подходит к каким-то достопримечательностям, то сервис сразу показывает информацию о нем или делает голосовые пояснения.

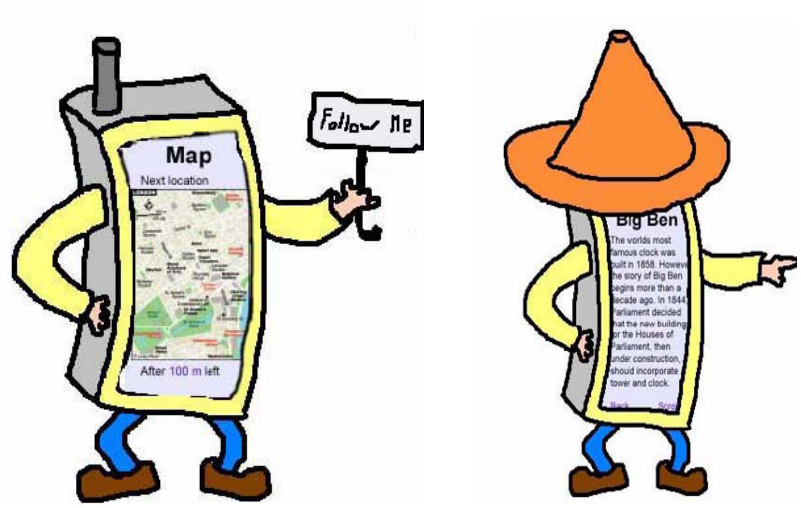

Рис. 2.4. Поддержка сервиса «Персональный Гид»

2.4.2. Организационная сетевая среда

Для удовлетворительной работы данного сетевого сервиса потребуются много источников информации и дополнительных сетевых сервисов. Откуда будет браться вся эта информация? Какие дополнительные сервисы потребуются? Кто будет ответственным за какую-то часть информации? Много участников должно совместно работать, чтобы обеспечить успешную работу данного сервиса. Можно выделить следующих участников (актеров), которые должны организовывать и распределять затраты по предоставлению данного сервиса:

1. *Сервисная организация «Персональный Гид»*. Данная организация будет создавать и поддерживать данный сервис. Она будет ответственной за функционирование данного сервиса. Пользователи этой сервисной организации будут ее клиентами и поэтому будут платить ей за предоставленные услуги. Данная организация должна

гарантировать правильную работу сервиса и предоставление правильной информации пользователю.

2. *Провайдер GIS* (Geographic Information System). Информация о месторасположении пользователя будет получаться с использованием GPS позиционирования или возможностей по позиционирования телекоммуникационной среды. В обоих случаях получаются координаты x и y. Эти координаты должны быть преобразованы в адреса, почтовые коды или место на карте.
3. *Провайдер информации о погоде*. Рассматриваемый сервис может использовать прогноз местной погоды. По крайней мере,сервис должен знать о том, прогнозируется ли дождь в ближайшее время. В связи с этим требуется провайдер информации о погоде.
4. *Провайдер информации об общественном транспорте*. Рассматриваемый сервис может советовать воспользоваться общественным транспортом. В связи с этим требуется знать, где расположены остановки или станции транспорта, какие существуют маршруты общественного транспорта, сколько времени потребуется на поездку и т. п. Имеется много сервисных провайдеров, предоставляющих такого вида информацию. Для каждого города, в котором доступен рассматриваемый сервис, должны быть заключены договора с провайдерами информации об общественном транспорте.
5. *Провайдеры туристической информации*. Для того, чтобы обеспечить работу рассматриваемого сервиса требуется информация о достопримечательностях, ресторанах, кинотеатрах и т. п., которые имеются в городе. Это может быть единый сервис, который уже доступен в городе или набор сервисов, предоставляемых различными провайдерами, с которыми должны быть заключены отдельные договора.
6. *Провайдер сервиса бронирования*. Рассматриваемый сервис должен иметь доступ к системе заказа билетов в кинотеатры и театры. В связи с этим, потребуется организация, которая предоставляет такую систему, например, посредством сети Интернет.
7. *Провайдер услуг сотовой связи*. Весь рассматриваемый сервис реализуется с использованием мобильных устройств пользователей. Эти мобильные устройства должны выполнять связь с сервером. Для этого обычно используются GPRS или UMTS соединения, которые предоставляются провайдером сети сотовой связи. Для того, чтобы гарантировать пользователю возможность использования рассматриваемого сервиса в любой стране потребуются соглашения о роуминге между провайдерами различных сетей. В случае, когда позиционирование выполняется с использованием UMTS се-

ти, то провайдер сотовой связи также будет предоставлять х-у координаты пользователя.

2.4.3. Техническая архитектура

Пользователи обращаются к сервису ПГ с помощью их мобильных устройств. Пользователи находятся в городе и мобильные устройства устанавливают соединение с местным оператором телекоммуникационной связи. Когда пользователь находится в других городах, то требуется соглашение о роуминге между местным оператором сотовых услуг и домашним оператором пользователя. С помощью GPRS или UMTS соединений пользователи могут получать доступ к сети Интернет. В сети Интернет они могут подключиться к основному серверу. Конечно, все это выполняется в результате простого выбора данного сервиса в списке Избранных ссылок.

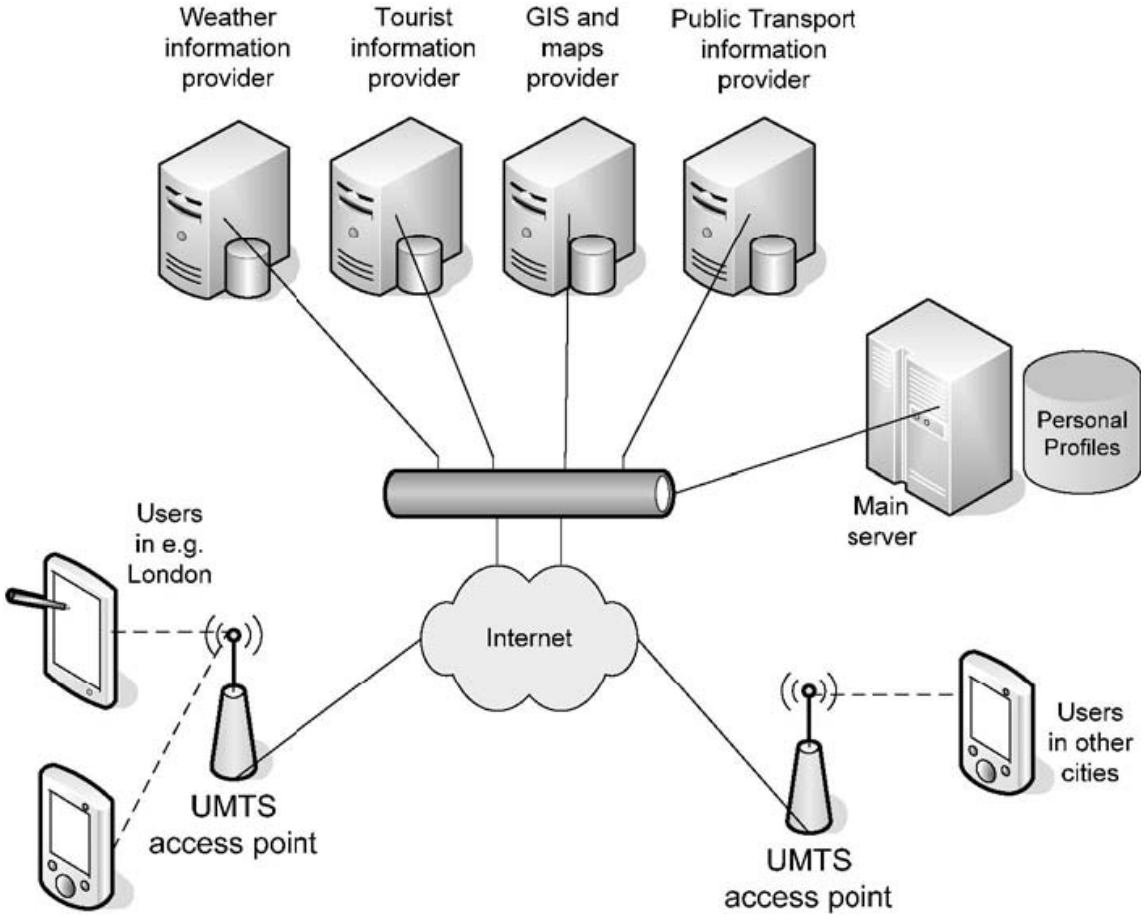

Рис. 2.5. Техническая архитектура, поддерживающая сервисную систему «Персональный Гид»

Основной сервер, на котором расположен данный сервис, также подсоединен к сети Интернет. С помощью сети Интернет данный сер-

вис соединяется со всеми дополнительными сервисами, которые требуются для формирования информации о предлагаемых пользователю маршрутах. Основной сервис также содержит базу данных с персональными профилями зарегистрированных пользователей.

Вопросы для самопроверки

1. Что такое мобильный сетевой сервис?
2. Что такое мобильный бизнес?
3. Предложите собственный вариант сетевого сервиса с использованием мобильных устройств и обоснуйте его выгоду для пользователей?
4. Как статьи затрат на разработку мобильных сервисных систем вы знаете?
5. Какие особенности имеют мобильные сервисные системы?
6. Какие возможности, с вашей точки зрения, можно добавить с сетевому сервису «Персональный Гид»?

3. ОСНОВЫ ПРОГРАММИРОВАНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

В настоящее время существует два основных варианта разработки программных систем для мобильных устройств (КПК, смартфоны и сотовые телефоны).

Первый вариант это создание клиентские приложения, обладающего графическим интерфейсом и сложной функциональностью, которые устанавливаются и выполняются на компьютере пользователя. Такие программы могут работать разным способом:

- локально (например, записные книжки, калькуляторы);
- совместно с программами, работающими на сервере, для организации взаимодействия, для получения доступа к базам данных, для выполнения сложных расчетов.

Взаимодействие с серверами может выполняться с использованием различных коммуникационных технологий. Это могут быть протоколы достаточно низкого уровня (например, сокеты) или протоколов более высокого уровня, например web-сервисы.

Другим вариантом создания приложений, работающих с сервером, является создание web-приложений. Web-приложения это прикладные программы, которые выполняются на стороне сервера, а для взаимодействия с пользователем используется только одну программу – web-браузер.

В данном пособии будут рассмотрены три варианта разработки и отладки мобильных приложений с использованием платформы .Net компании Microsoft: локальные приложения; локальные приложения, взаимодействующие с web-сервисами и мобильные web-приложения.

3.1. Отладка мобильных приложений

В состав Visual Studio.Net входит несколько вспомогательных программ, которые позволяют выполнять отладку мобильных приложений без использования реальных устройств.

3.1.1. Эмуляторы мобильных устройств

При создании приложений для КПК и смартфонов необходимо проверять работу написанной программы на устройстве, которое сильно отличается от настольного компьютера. Когда создается стандартное Windows-приложение, то можно сразу увидеть его работу, запустив соответствующий исполняемый файл. Написав программу для мобильных

устройств, необходимо протестировать ее на соответствующем устройстве, так как настольный компьютер здесь уже не поможет. Но даже если разработчик еще не приобрел карманный компьютер или смартфон под управлением Windows Mobile, то он все равно может тестировать свои приложения. В этом случае надо проверять их работоспособность на специальных эмуляторах.

Следует отметить, что в некоторых случаях эмулятор все-таки не сможет выполнить эту задачу. Например, он не поможет проверить работу кода, который использует возможности инфракрасной связи. И, тем не менее, эмулятор является очень мощным и удобным инструментом для отладки приложений.

Надо сказать, что качество и возможности эмулятора постоянно улучшаются и совершенствуются. Эмулятор, поставляемый с Visual Studio, имеет улучшенную поддержку общих папок, программы синхронизации ActiveSync и последовательных портов. Также эмулятор поддерживает альбомную и книжную ориентацию. Кроме этого можно дополнительно скачать версии эмуляторов с поддержкой разных языков.

3.1.2. Запуск эмуляторов на выполнение

Таким образом, при написании своей программы у разработчика есть возможность выбирать, где тестировать свой код. Как правило, программу сначала проверяют на эмуляторе. Это позволяет быстро исправить ошибки и устранить недочеты. А уже окончательную версию программы можно и нужно проверить на реальном устройстве.

Рассмотрим вариант запуска эмулятора и его настройки. Сначала требуется создать или открыть проект, предназначенный для мобильного устройства. После выполнения команды меню Debug > Start Debugging среда разработки Visual Studio отображает диалоговое окно Deploy (рис. 3.1).

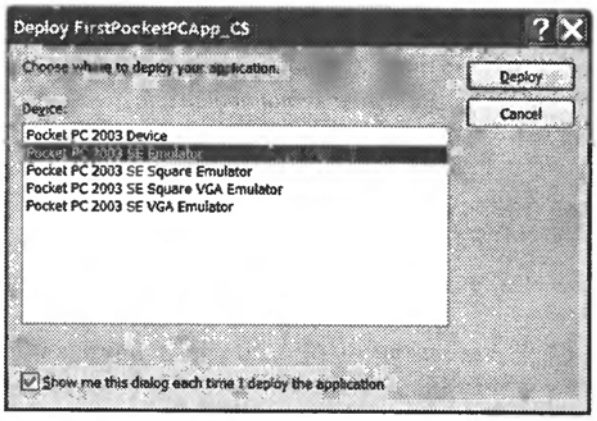

Рис. 3.1. Запуск эмулятора

В диалоговом окне отображается список, в котором содержатся одно реальное устройство и четыре эмулятора разных типов устройств.

Нужно выбрать любой эмулятор из списка. Стандартным выбором в данном *случае* является значение Pocket PC 2003 SE Emulator. Нужно выделить строку с выбранным эмулятором и нажать кнопку «Deploy». Через несколько секунд на экране компьютера появится эмулятор карманного компьютера, в котором будет запущено выбранное приложение. Программист может работать с тестируемой программой так же, как и на реальном устройстве. Кроме того, можно оставить в покое программу и запустить любое приложение, которое есть на этом эмуляторе.

Замечание: Списки эмуляторов на каждом компьютере разработчика могут различаться, так как можно скачать и установить дополнительные эмуляторы. Когда будут рассматриваться примеры для устройств под управлением Windows Mobile 5.0, диалоговое окно будет содержать уже другие эмуляторы.

После того как тестирование программы будет завершено, необходимо остановить выполнение программы при помощи команды меню «Stop debugging» в среде разработки. При этом не стоит закрывать само окно эмулятора. Если оставить эмулятор в рабочем состоянии, то это позволит потратить меньше времени на повторную загрузку эмулятора при следующей отладке программы.

Замечание: Если разрабатываемая программа имеет код для закрытия приложения this.Close(), то режим отладки автоматически остановится и выполнять команду меню Stop debugging не понадобится.

3.1.3. Настройка эмуляторов

Для настойки эмулятора следует выполнить команду меню Tools->Options. В открывшемся диалоговом окне «Options» надо выбрать строку «Device Tools», а в ней активировать пункт «Devices». Затем в списке «Devices» надо выбрать элемент Pocket PC 2003 SE и нажать кнопку «Properties» (рис. 3.2).

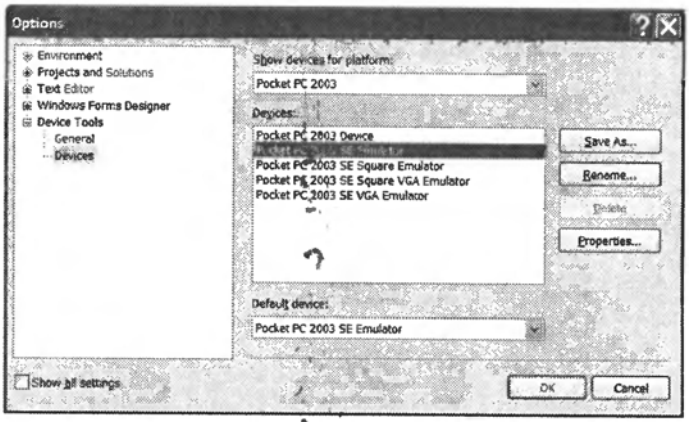

Рис. 3.2. Окно настроек эмулятора

На экране появится новое диалоговое окно Pocket PC 2003 SE Properties (рис. 3.3). Обратите внимание на то, что по умолчанию программа устанавливается в папку «Program Files».

После ознакомления со свойствами эмулятора нужно закрыть все диалоговые окна и вернуться в главное окно среды разработки. Там надо выполнить команду меню Tools->Device Emulator Manager. На экране откроется новое диалоговое окно, в котором будут перечислены все имеющиеся эмуляторы.

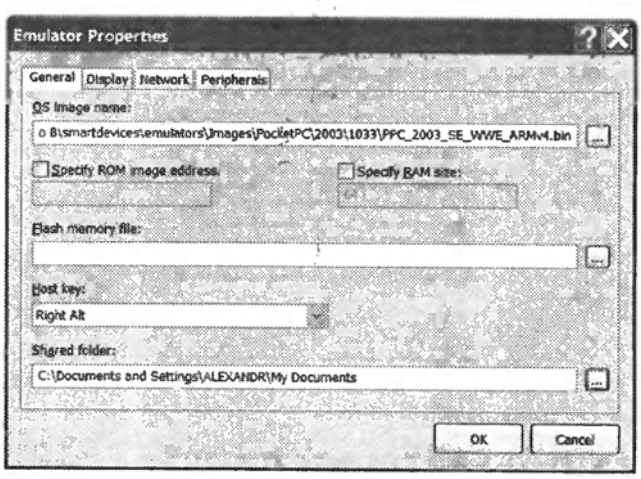

Рис. 3.3. Окно свойств эмулятора

Надо выбрать из списка элемент Pocket PC 2003 SE Emulator, а затем выполнить команду меню Actions->Connect. Менеджер эмуляторов загрузит выбранный эмулятор. На экране появится специальный значок, который сигнализирует об установленном соединении. Затем нужно выполнить команду меню Actions->Cradle. Если операция пройдет успешно, то значок состояния эмулятора изменится. Это означает, что эмулятор КПК соединен с виртуальной док-станцией (устройства позволяющего соединять мобильных устройств с настольными персональными компьютерами). Теперь можно синхронизировать данные с помощью программы синхронизации ActiveSync. По завершении операции нужно выполнить команду меню Actions->Uncradle.

3.1.4. Изменение ориентации экрана

Эмулятор позволяет легко менять ориентацию экрана. Достаточно перейти на вкладку Display после выполнения команды меню File->Configure и выбрать нужный режим в разделе Orientation (рис. 3.4).

Если выбрать соответствующее значение для поворота экрана, то эмулятор повернет изображение устройства (но не экрана) на 90° (рис. 3.5).

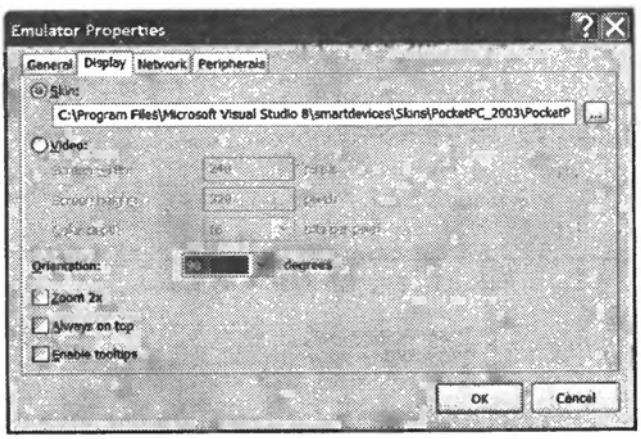

Рис. 3.4. Настройка ориентации экрана

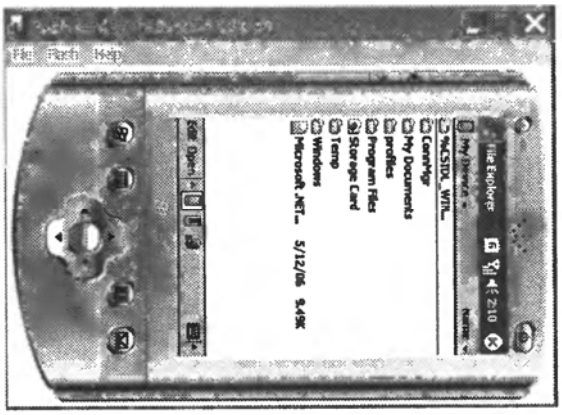

Рис. 3.5. Вращение устройства

3.1.5. Выход в сеть Интернет с помощью эмулятора

Для проверки мобильных web приложений также можно использовать эмуляторы. Если компьютер, на котором установлен эмулятор, имеет соединение с Интернетом, то можно подключить к Сети и сам эмулятор. Настройка не очень сложна, и все этапы приведены в следующем списке:

1. Создать новый проект в Visual Studio.Net.
2. Запустить программу ActiveSync. Возможно, она неактивна, и ее пиктограмма располагается в области уведомлений. В этом случае надо щелкнуть правой кнопкой мыши на этом значке и выполнить команду контекстного меню «Открыть Microsoft ActiveSync».
3. Вернуться в среду разработки Visual Studio и выполнить команду меню Tools->Device Emulator Manager. На экране появится диалоговое окно «Device Emulator Manager».
4. Щелкнуть правой кнопкой мыши на соответствующем эмуляторе и выполнить команду контекстного меню Connect. На экране появится соответствующий эмулятор.

5. Вернуться в диалоговое окно «Device Emulator Manager» и снова щелкнуть правой кнопкой на выбранном ранее эмуляторе, а затем выполнить команду Cradle.
6. В диалоговом окне «Device Emulator Manager» у выбранного эмулятора появится значок, показывающий, что эмулятор теперь подключен к системе настольного компьютера.
7. Автоматически появится сообщение от «Microsoft ActiveSync» о том, что установлено соединение (рис. 3.6).

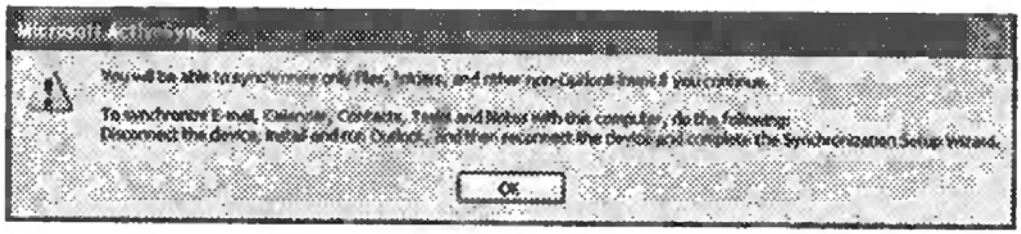

Рис. 3.6. Сообщение ActiveSync

8. В этом окне нужно нажать кнопку ОК.
9. На экране появится окно «Мастер синхронизации» (рис. 3.7).

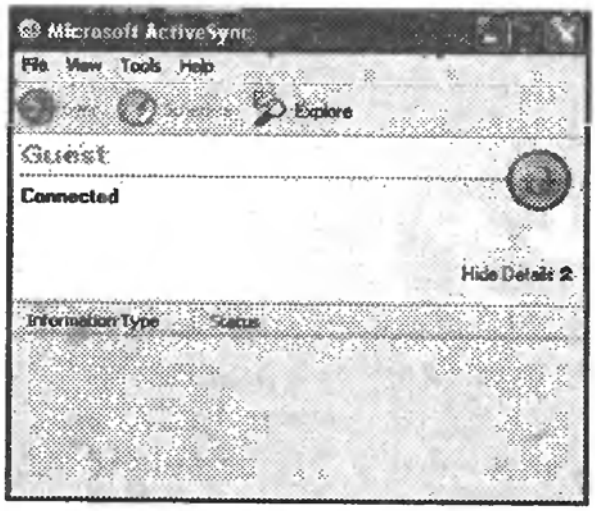

Рис. 3.7. Окно ActiveSync

10. Так как сейчас синхронизация не нужна, то следует нажать кнопку Cancel.
11. Появится основное окно программы «Microsoft ActiveSync», сигнализиру-ющее, что установлено соединение с компьютером.
12. Закрыть окно программы «Microsoft ActiveSync». Программа продолжает работать в фоновом режиме. В области уведомлений должна отображаться зеленая пиктограмма.

13. В очередной раз вернуться в окно программы Device Emulator Manager и закрыть его. Программа также продолжает работать в фоновом режиме, а ее пиктограмма тоже появится в области уведомлений.
14. Настало время настройки эмулятора для доступа в Интернет. Нужно щелкнуть правой кнопкой на зеленом значке ActiveSync и выполнить команду контекстного меню «Открыть Microsoft ActiveSync». Затем надо выполнить команду меню File->Connection Settings и выбрать режим «This computer is connected to The Internet», после чего останется только нажать кнопку ОК.
15. В эмуляторе надо нажать кнопку Start и щелкнуть на пиктограмме Internet Explorer. В результате будет запущен стандартный браузер. В адресной строке можно указать URL любого существующего сайта. Эмулятор должен загрузить выбранный сайт.

Теперь данный компьютер соединен с Интернетом через эмулятор. Это позволит отлаживать web-приложения.

3.2. Программирование локальных мобильных приложений

3.2.1. Разработки первого приложения

Прежде всего, нужно запустить среду разработки Microsoft Visual Studio.Net и создать новый проект. Первые различия в процессе разработки можно увидеть уже на этой стадии. Если для создания обычных приложений надо было выбрать раздел Windows, то на этот раз необходимо выбрать раздел «Smart Device».

В этом разделе содержатся подразделы, которые отвечают за создание приложений для КПК, смартфонов и устройств под управлением операционной системы Windows CE. Здесь нужно указать язык программирования C#, перейти в раздел Smart Device и выбрать подраздел Pocket PC 2003 (рис. 3.8).

В выбранном подразделе присутствуют несколько шаблонов для реализации различных задач. Обычно используется шаблон «Device Application». После того как будет выбран шаблон для приложения, требуется изменить имя проекта. По умолчанию используется название `DeviceApplication1`, но наше первое приложение получит имя `FirstMobileApp`. После нажатия кнопки ОК откроется окно среды разработки с необычным видом формы. Если при программировании программ для настольных компьютеров отображается только форма, то в данном случае на экране будут показаны не только форма, но и внешний вид мобильного устройства. Разработчик может отключить отображение устройства, оставив на экране только форму. Для этого нужно щелкнуть правой кнопкой мыши на форме и в появившемся кон-

текстном меню выбрать пункт «Show Skin». Повторный выбор этого пункта вернет на экран стандартный вид формы.

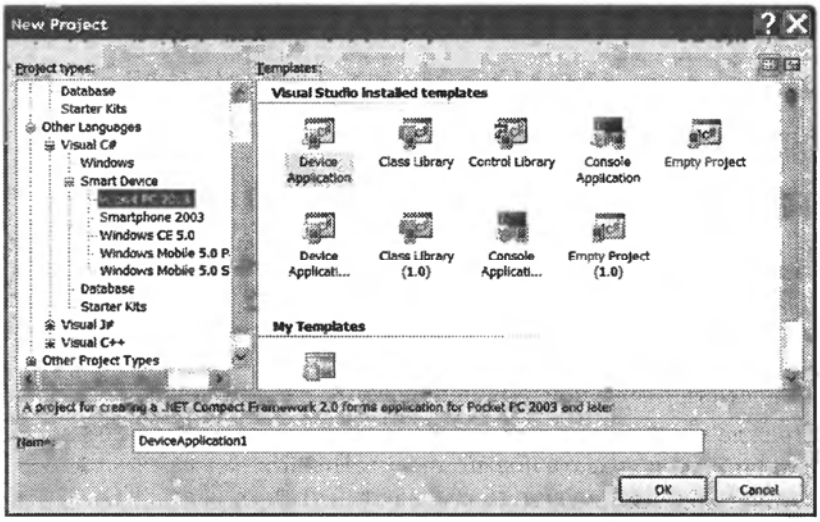

Рис. 3.8. Выбор типа платформы

Обычно в качестве первого примера создается стандартная программа, которая выводит приветствие на экран. На форме надо расположить кнопку `Button` и элемент `Label` для отображения надписи. Также потребуется написать код для обработчика события `Click` созданной кнопки. Код этой программы приведен в листинге 3.1.

Листинг 3.1. Программа «Здравствуй, мир!»

```
private void butSayHello_Click (object sender, EventArgs e)
{
     lblHello.Text =
       "Здравствуй, мир!";
}
```

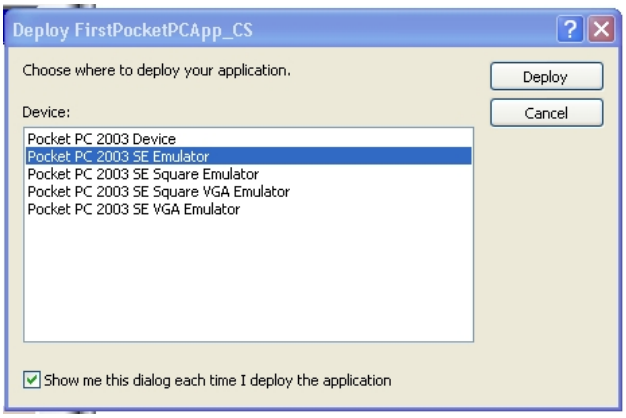

Рис. 3.9. Диалоговое окно Deploy

Теперь можно запустить проект при помощи команды «Start Debugging» или клавиши быстрого вызова F5. На экране появится диало-

говое окно `Deploy` (рис. 3.9) для выбора устройства, с помощью которого будет выполняться разработанная программа.

В основном списке этого окна перечислены устройства, на которых может выполняться написанная программа. Проверять работу приложения можно как на реальном устройстве, так и при помощи эмулятора. Как правило, при отладке программы используют эмуляторы и только в финальной части тестируют программу на реальном устройстве.

Для выполнения примеров из пособия сначала нужно запустить эмулятор. Для этого можно выбрать любой эмулятор из предложенного списка. Чаще всего применяется эмулятор Pocket PC 2003 SE. После выбора соответствующего значения в списке нужно нажать кнопку `Deploy`. Сначала на экране монитора будет отображен сам эмулятор (рис. 3.10), а спустя некоторое время в эмуляторе будет запущена созданная программа.

Мышью можно щелкнуть на кнопке с надписью «Поздороваться». В результате на форме появится строка «Здравствуй, мир!» (рис. 3.11).

Рис. 3.10. Первый запуск эмулятора

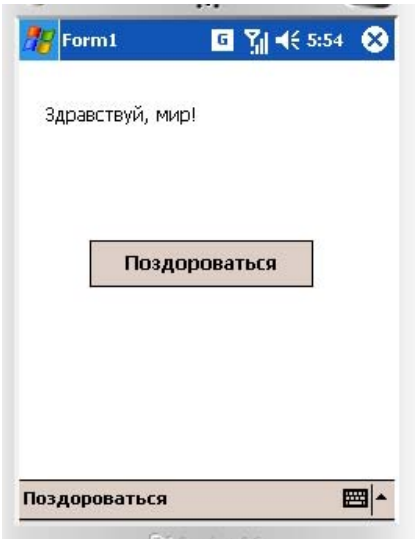

Рис. 3.11. Первое мобильное приложение

Размеры и позиция формы по умолчанию любая форма занимает весь экран. При этом у формы будет всего одна системная кнопка минимизации вместо привычных трех кнопок свертывания, восстановления и закрытия формы. Причем кнопка минимизации в виде крестика очень похожа на кнопку закрытия формы в обычных настольных приложениях. Стандартная модель поведения программ на карманном компьютере устроена таким образом, что когда пользователь нажимает кнопку с крестиком, то он сворачивает программу, а не закрывает ее. Об этом говорит значение `True` свойства формы `MinimizeBox`. Если

для этого свойства задать значение `False`, то вместо кнопки с крестиком будет отображаться кнопка `OK`. При нажатии на эту кнопку программа завершит свою работу.

По умолчанию любая форма занимает весь экран. Ее верхний левый угол находится в точке с координатами (0, 26). Если попробовать вручную изменить значения свойства `Location`, то среда разработки проигнорирует эти попытки и вернет значения. Что же касается размеров формы, то при желании все же можно изменить высоту и ширину формы.

Программная панель ввода

Большинство карманных компьютеров не имеют встроенных клавиатур для ввода информации. Вместо клавиатуры в этом случае используется специальная программная панель ввода «Software Input Panel» (SIP), которая позволяет вводить текст (рис. 3.12).

Для работы с виртуальной клавиатурой в .NET Compact Framework используется класс `InputPanel`. Так как панель ввода находится на панели задач, то необходимо, чтобы панель задач была видимой. А ранее уже говорилось что, если форма не имеет меню, то панель задач будет невидима. В результате при попытке создания экземпляра класса `InputPanel` на форме, не имеющей меню, будет отображено сообщение об ошибке.

Стилус вместо устройства «мышь»

В карманных компьютерах роль устройства «мышь» выполняет стержень из пластика, называемый ***стилусом***. Конечно, у стилуса вообще нет кнопок. Вместо кнопок в карманных компьютерах применяется технология «Tap-and-Hold». Для выделения элемента управления пользователь должен точно попасть в него кончиком стилуса. По аналогии с мышью, можно легко щелкнуть по экрану (`Click`), а можно нажать на экран и удерживать стилус на месте (`Press`).

Рис. 3.12. Активированная панель ввода SIP

3.2.2. Программирование графического интерфейса

Несмотря на свою схожесть, .NET Compact Framework уступает в функциональности базовой библиотеке .NET Framework. Это относится и к элементам управления. Обычные элементы управления, такие, как кнопки, списки и текстовые поля присутствуют и в мобильной версии.

Нужно помнить, что даже поддерживаемые элементы управления имеют порой ограниченные возможности. Чтобы узнать, какие свойства, методы или события не поддерживаются элементом управления, нужно запустить справочную систему, найти нужный класс и просмотреть все члены класса. Если нужное свойство поддерживается в .NET Compact Framework, то у его описания будет присутствовать значок мобильного устройства.

Элемент управления `Form`

Элемент `Form` является контейнером для элементов управления и является рабочей площадкой для создания пользовательского интерфейса программы. Класс `Form` имеет несколько свойств, которые могут различаться в зависимости от выбранной целевой платформы.

- **Свойство `FormBorderStyle`** – определяет стиль формы. По умолчанию используется стиль `FormBorderStyle.FixedSingle`. При этом форма заполняет все рабочее место экрана, и пользователь не может изменять размеры формы или перемещать ее по экрану. При установке значения `FormBorderStyle.None` создается форма без рамки и заголовка. В этом случае можно изменять размеры и расположение формы программно, но пользователь по-прежнему не может манипулировать формой.
- **Свойство `ControlBox`** – отвечает за отображение контейнера для элемента управления. Если свойство `ControlBox` имеет значение True, то контейнер будет отображаться. В противном случае он на экран не выводится. Для устройств Pocket PC подобный контейнер может содержать только одну кнопку.
- **Свойства `MinimizeBox` и `MaximizeBox`.** В приложениях для Pocket PC форма может содержать только одну кнопку. Она отвечает либо за минимизацию формы, либо за ее закрытие. Разработчик может управлять внешним видом кнопки при помощи свойства `MinimizeBox`. Если оно имеет значение `True`, то кнопка при нажатии будет сворачивать форму. Значение `False` позволяет создавать кнопку закрытия формы. Значение свойства `MaximizeBox` игнорируется системой.
- **Свойство `WindowsState`** – определяет состояние окна при первоначальной загрузке. Разработчик может использовать значения `FormWindowState.Normal` и `FormWindowState.Maximized`. Если свойство имеет значение `FormWindowState`. Normal то форма заполняет весь экран, за исключением нижней полоски меню и верхней полоски системного меню `Start` (Пуск). При использо-

вании значения `FormWindowState.Maximized` форма заполняет экран полностью, скрывая системное меню `Start` (Пуск), но при этом нижняя полоса меню остается видимой.

- **Свойство `Size`** – позволяет задавать размеры формы. Это свойство игнорируется, если свойство `FormBorderStyle` имеет значение `FixedSingleProperty`.
- **Свойство `Location`** – задает координаты верхнего левого угла формы. Но так как форма обычно заполняет весь экран, то в большинстве случаев это свойство не используется.

Элемент управления Button

Для создания обычной кнопки используется класс `System.Windows.Forms.Button`. Эта кнопка обладает всеми основными функциями, которые есть у такого же класса в полной версии .NET Framework. Кнопка предназначена для обработки нажатия стилуса на соответствующую область экрана. В этом случае возникает событие `Click`. Прогроммный код, показанный в листинге 3.2, является обработчиком этого события. Он выводит текущее время в текстовое поле после нажатия на кнопку с надписью «Узнать время».

Листинг 3.2.

```
private void butGetTime_Click(object sender, EventArgs e)
{
     txtCurTime.Text = DateTime.Now.ToLongTimeString();
}
```

На рис. 3.13 показан интерфейс приложения в момент нажатия на кнопку. Текст на кнопке может быть только однострочным. Если он не помещается на кнопке, то будет обрезан. Поэтому нужно быть очень осторожным при выборе текста для кнопки. Функциональность элемента управления `Button` очень сильно урезана по сравнению с полной версией .NET Framework. В частности, у данного элемента нет свойств `Image` и `ImageList`, которые применяются для отображения на кнопке графики.

Рис. 3.13. Результат нажатия на кнопку

Элемент управления TextBox

В предыдущем примере дата отображалась в текстовом поле. Это поле создается при помощи класса `TextBox`, который позволяет вводить текст. Данный элемент поддерживает такие стандартные свойства, как `BackColor` и `ForeColor`. Событие `Click` элементом `TextBox` не

поддерживается, но разработчик может воспользоваться событиями Keypress, KeyUp и KeyDown. Следует отметить особенность этого элемента. Несмотря на то, что класс TextBox поддерживает свойство PasswordChar, при вводе пароля на экране всегда будет использоваться символ звездочки.

Элемент управления Label

В рассмотренном примере также использовался элемент Label для отображения текстовой строки. Как правило, надпись используется для отображения некоторого текста, который пользователь не может изменить. Сама отображаемая строка задается при помощи свойства Text. Текст на экране можно выравнивать с помощью свойства TextAlign. Разработчик может использовать значения TopLeft, TopCenter и TopRight. При изменении текста в метке инициируется событие TextChanged. При создании элемента нужно следить за длиной отображаемой строки. Если текст слишком большой и не помещается в пределах элемента, то он попросту обрезается.

Элемент управления RadioButton

Элемент управления RadioButton позволяет создавать переключатели, объединенные в группы. Вся группа переключателей должна располагаться в контейнере. Примером такого контейнера может служить сама форма, но чаще используется элемент Panel. Когда пользователь выбирает один переключатель, то остальные переключатели в контейнере автоматически переводятся в выключенное состояние. Приложение может иметь несколько групп элементов RadioButton. В любом случае группы переключателей не зависят друг от друга.

При изменении состояния переключателя в классе RadioButton инициируются события Click и CheckedChanged. Событие Click возникает, когда пользователь щелкает стилусом на самом переключателе. Событие CheckedChanged возникает, когда состояние элемента RadioButton меняется программно или в результате действий пользователя. Событие Click не инициируется, когда свойство CheckedChanged меняется программно.

Для демонстрации примера работы с элементом RadioButton можно создать аналог популярной телеигры «Кто хочет стать миллионером?». На экране будет отображаться вопрос, а пользователь должен выбрать из представленных вариантов единственный правильный ответ. Код, реализующий основную функциональность приложения, приведен в листинге 3.3.

Листинг 3.3.

```
private void radClubl_CheckedChanged(object sender,
     EventArgs e)
{
   if (this.radClubl.Checked)
     MessageBox.Show ("Увы, вы проиграли". "Ошибка!");
}
private void radClub2_CheckedChanged(object sender,
     EventArgs e)
{
   if (this.radClub2.Checked)
     MessageBox.Show ("Поздравляю! Вы выиграли миллион!",
          "Миллион!");
}
private void radClub3_CheckedChanged(object sender,
     EventArgs e)
{
   if (this.radClub3.Checked)
     MessageBox.Show ("Увы, вы проиграли", "Ошибка!");
}
private void radClub4_CheckedChanged( object sender,
     EventArgs e)
{
   if (this.radClub3.Checked)
          MessageBox.Show ("Увы. вы проиграли". "Ошибка!");
}
```

На рис. 3.14 показан внешний вид этого приложения. В полной версии .NET Framework в качестве контейнера для переключателей часто используется элемент `GroupBox`, который на данный момент не поддерживается в библиотеке .NET Compact Framework. Также не поддерживаются некоторые свойства, к которым относятся `Appearance`, `Image` и `ImageList`.

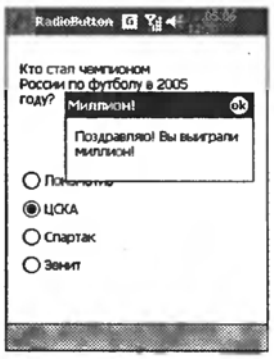

Рис. 3.14. Демонстрация работы независимых переключателей

Элемент управления CheckBox

Элемент управления `CheckBox` позволяет создавать независимый переключатель в виде флажка. Элемент `CheckBox` имеет свойство `CheckState`, позволяющее определить состояние переключателя. Программист может использоваться значения `Unchecked`, `Checked` и `Indeterminate`. Значение `Unchecked` свидетельствует о том, что флажок в переключателе не взведен. Если переключатель все же включен, то используется значение `Checked`.

Состояние Indeterminate используется, когда для свойства ThreeState элемента CheckBox установлено значение True. Если свойство CheckState имеет значение Indeterminate, то элемент окрашен серым цветом, но, тем не менее, считается помеченным. При этом пользователь не может изменить состояние переключателя. Также элемент не распознает событие Click, если свойство AutoCheck имеет значение False. Для этого свойства нужно задать значение True, чтобы пользователь мог пользоваться стилусом для работы с переключателем.

Элемент управления ComboBox

Элемент управления ComboBox позволяет создавать поле со списком выбора. Благодаря своей компактности этот элемент управления хорошо подходит для тех задач, когда требуется экономить место на экране. Поле со списком выглядит как обычное текстовое поле TextBox со стрелкой, которая расположена в правой части поля. Когда пользователь щелкает по стрелке, то открывается список с предварительно заданными элементами. Когда пользователь выбирает определенный пункт списка или снова щелкает по стрелке, то список снова сворачивается.

Добавлять текстовые элементы в ComboBox можно как в режиме проектирования, так и программно во время работы программы. В листинге 3.4 показан пример добавления пунктов программным путем.

Листинг 3.4.

```
comboBox1.Items.Add("Мурзик");
comboBox1.Items.Add("Барсик");
comboBox1.Items.Add("Рыжик");
```

Для этого нужно вызвать метод Add в свойстве коллекции Items элемента ComboBox. Отдельные пункты можно удалять с помощью метода Remove, а чтобы удалить все пункты сразу, применяется метод Clear. Приведенный пример показывает, как можно добавить три строки в элемент ComboBox с именем comboBox1.

Чтобы узнать, какой элемент выбрал пользователь, применяется свойство SelectedIndex или SelectedItem. Свойство SelectedIndex возвращает порядковый номер выбранного пункта. Этот номер можно использовать для доступа к выбранному пункту при помощи свойства Items. Следует помнить, что нумерация элементов начинается с нуля. Пример работы со свойством SelectIndex приведен в листинге 3.5. Также в этом листинге показано, как можно получить доступ к выбранному пункту при помощи свойства SelectedItem.

Листинг 3.5.

```
// Получим выделенный пункт с помощью SelectedIndex
```

```
string selItem =
   (string) cmbCats.Items[cmbCats.SelectedIndex];
MessageBox.Show(selItem);
// Второй способ – получим пункт с помощью Selectedltem
string selItem = cmbCats.SelectedItem.ToString();
MessageBox.Show(selItem);
```

В полной версии .NET Framework у элемента `ComboBox` для свойства `DropDownStyle` можно задавать значения `Simple`, `DropDownList` или `DropDown`. В .NET Compact Framework значение Simple не используется. По умолчанию в .NET Compact Framework применяется значение `DropDownList`, тогда как в полной версии .NET Framework по умолчанию используется стиль DropDown. Также не поддерживаются многие методы из основной версии библиотеки. В .NET Compact Framework 2.0 у поля со списком появилась поддержка методов BeginUpdate и EndUpdate, которые позволяют избежать мерцания при загрузке большого числа элементов.

Замечание: Внешний вид и поведение элемента ComboBox в смартфонах немного отличается от аналогичных элементов в КПК. Более подробно об отличиях будет рассказано в соответствующей главе.

Элемент управления ListBox

Элемент `ComboBox` подходит для приложений с ограниченными пространствами формы, а список `ListBox` можно использовать, если на экране достаточно места для отображения всех пунктов списка. Список `Listbox` сразу показывает все имеющиеся элементы списка, при необходимости добавляя вертикальную полоску прокрутки, если все элементы списка не могут быть отображены одновременно. Элементы `ComboBox` и `ListBox` имеют почти одинаковый набор свойств и методов. В листинге 3.6 показано, как можно программно добавить несколько строк в список `ListBox`.

Листинг 3.6.

```
lstFruit.Items.Add("Яблоко");
lstFruit.Items.Add("Груша");
lstFruit.Items.Add("Слива");
lstFruit.Items.Add("Персик");
```

Свойство `SelectedIndex` содержит порядковый номер выбранного элемента списка. Если указать этот индекс в коде приложения, то выбранный элемент будет немедленно выделен в списке соответствующим цветом. Если никакой элемент не выбран, то свойство `SelectedIndex` имеет значение –1. Также класс поддерживает свойство `SelectedItem`, которое соответствует одноименному свойству класса `ComboBox`.

Из часто используемых свойств элемента `ListBox` в полной версии .NET Framework можно выделить свойство `MultiColumn`, которое не поддерживается в .NET Compact Framework. В нем отсутствует горизонтальная полоска прокрутки, даже если строки текста не умещаются в списке полностью. Также не поддерживается многострочное выделение, поэтому пользователь может выбрать только один элемент списка.

Элемент управления ProgressBar

Элемент управления `ProgressBar` предназначен для индикации процесса выполнения какой-либо операции. Как правило, данный элемент активно используется при выполнении долгих операций, чтобы

пользователь получил иллюзию контроля над работой приложения.

Чаще всего разработчик работает со свойствами `Minimum`, `Maximum` и `Value`. Свойства `Minimum` и `Maximum` задают минимальное и максимальное значения свойства `Value`. А свойство `Value` определяет текущее значение индикатора. Обычно, данный элемент отображается в момент начала долгой операции, а после ее завершения делается невидимым с помощью метода `Hide` или свойства `Visible`.

Для демонстрации работы индикатора прогресса было создано приложение, которое позволит отследить время варки яиц вкрутую. Предположим, что для варки достаточно трех минут. Нужно положить яйца в воду и запустить таймер. По истечении трех минут приложение должно отобразить соответствующее сообщение. Основной код приложения приведен в листинге 3.7.

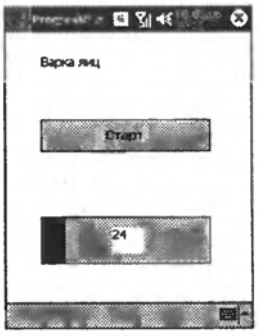

Рис. 3.15. Индикатор прогресса

Листинг 3.7.

```
private void tmrCook_Tick(object sender,
EventArgs e)
{
   if(this.progressBar1.Value < this.progressBarl.Maximum){
      this.progressBar1.Value += 1;
      lblCounter.Text = this.progressBarl.Value.ToString();
   }
   if(this.progressBarl.Value >= this.progressBar1.Maximum){
      MessageBox.Show("Яйца сварились !");
      this.progressBarl.Value = 0;
      lblCounter. Text = "0";
   }
}
private void butStart_Click( object sender, EventArgs e)
{   tmrCook. Enabled = true; }
```

На рис. 3.15 показан внешний вид приложения в момент отсчета времени.

Элемент управления StatusBar

Строка состояния выглядит как небольшая полоска в нижней части приложения, в которой отображается текстовая информация для пользователя. Этот элемент интерфейса реализуется при помощи элемента `StatusBar`. Чтобы изменить текст в элементе `StatusBar`, достаточно присвоить новое значение свойству `Text`. На рис. 3.16 показан внешний вид приложения в тот момент, когда пользователь нажимает на кнопку, а в листинге 3.8 приведен пример кода, который меняет текст в строке состояния.

Рис. 3.16. Пример работы со строкой состояния

Листинг 3.8.

```
private void butClickMe_Click
     (object sender, EventArgs e)
{
     this.statusBarl.Text =
          "Вы нажали на кнопку";
}
```

Элемент управления ToolBar

Элемент управления `ToolBar` позволяет создавать собственную панель инструментов. Во многих случаях использование панели инструментов может принести разработчику больше выгод, чем применение меню. Следует учитывать, что панель инструментов позволяет использовать изображения, что делает работу с этим элементом удобным и наглядным. В .NET Compact Framework элемент `ToolBar` не может содержать текст. Все инструменты маркируются только при помощи графических изображений. Изображения связываются с элементом при помощи класса `ImageList`. В приложениях для КПК панель инструментов всегда располагается в нижней части экрана справа от пунктов меню (рис. 3.17).

Рис. 3.17. Панель инструментов в приложении

Чтобы добавить в приложение элемент `ToolBar`, нужно сначала переместить на форму элемент управления `ImageList`. Значок `ImageList` появится в нижней части окна «Form Designer» рядом с элементом

mainMenu. В окне редактора свойств Properties нужно выбрать свойство Images и нажать кнопку редактирования. В результате этих действий будет открыто диалоговое окно «Image Collection Editor».

В этом окне следует добавить изображения, предназначенные для панели инструментов. Рекомендуется использовать картинки размером 16×16 пикселей, чтобы при их отображении не возникло искажений. Выбранные изображения включаются в состав приложения, и их не придется отдельно поставлять вместе с программой.

Теперь на форму надо перенести элемент ToolBar. В его свойстве ImageList надо указать имя добавленного ранее элемента ImageList. В рассматриваемом примере использовалось имя по умолчанию Image-List1. Затем надо перейти к свойству Buttons и активировать окно редактора «ToolBarButton Collection Editor». Так как было добавлено три изображения для кнопок, то нужно три раза нажать кнопку Add. Для каждой добавленной кнопки следует задать свойство ImageIndex. При необходимости разработчик может изменить стиль отображения кнопок. По умолчанию используется стиль PushButton, который создает обычные кнопки. Если для кнопки задать стиль DropDownButton, то при ее нажатии будет отображаться меню или другое окно. Справа от кнопки на панели инструментов отображается стрелка.

Стиль Separator позволяет создавать разделители между кнопками на панели управления. Стиль ToggleButton позволяет создавать переключаемую кнопку. При первом нажатии она переходит в активированное состояние, в котором и остается до тех пор, пока кнопку не нажмут повторно. В результате этих действий, панель инструментов добавлена к форме, хотя еще не было написано ни одной строчки кода.

При щелчке на кнопках элемента ToolBar возникает событие Button-Click. В листинге 3.9 приведен код, обрабатывающий нажатие первых двух кнопок.

Листинг 3.9.

```
private void tool Bar1_ButtonClick (object sender,
     ToolBarButtonClickEventArgs e)
{
   if(e.Button == this.toolBarButton1)
   {
      MessageBox.Show("Вы выбрали первую кнопку");
   }
   else if(e.Button == this.toolBarButton2)
   {
      MessageBox.Show("Вы выбрали вторую кнопку");
   }
}
```

Элемент управления MainMenu

Меню является одним из самых важных элементов графического интерфейса Windows-приложений. Не являются исключением и программы для мобильных устройств. По умолчанию на форме уже присутствует элемент `MainMenu`. Но при добавлении в проект новой формы на ней меню не появляется, и его нужно добавить вручную.

Следует помнить, что в приложениях для Pocket PC меню располагается в нижней части окна программы, тогда как в приложениях для обычных компьютеров меню располагается в верхней части окна. Если в программе одновременно присутствуют меню и панель инструментов, то они будут отображаться вместе. Но меню будет прижато к левой границе окна, а панель инструментов – к правой. Пример работы с меню приведен в листинге 3.10.

Листинг 3.10.

```
private void mnuAbout_Click (object sender, EventArgs e)
{
   MessageBox.Show("Это моя программа!");
}
private void mnuExit_Click (object sender, EventArgs e)
{ this.Close();}
```

Элемент управления ContextMenu

Элемент `ContextMenu` позволяет создавать контекстные меню для других элементов интерфейса. Этот элемент очень похож на элемент управления `MainMenu`. Но если `MainMenu` всегда связан с формой приложения, то `ContextMenu` можно связать с любым элементом формы. Так как в КПК не используется мышь, то вызов контекстного меню вызывается операцией tap-and-hold вместо привычного щелчка правой клавишей мыши.

Замечение: Если используетеся эмулятором, то для имитации tap-and-hold нужно щелкнуть левой кнопки мыши и не отпускать ее некоторое время.

Чтобы добавить элемент `ContextMenu` в приложение, нужно сначала переместить его значок на форму. Он появится в нижней части редактора Form Designer, там же, где и элемент `MainMenu`. Но на самом деле во время выполнения программы контекстное меню будет отображаться рядом с выбранным элементом интерфейса. Также контекстное меню можно создавать программно во время запуска приложения.

При вызове контекстного меню инициируется событие `Popup`. Когда пользователь выбирает какой-то пункт меню, то возникает событие `Click`. Чтобы привязать созданное контекстное меню к конкретному элементу интерфейса, нужно выбрать его на форме и в свойстве `ContextMenu` указать созданное контекстное меню.

Элемент управления Timer

Элемент `Timer` позволяет выполнять некоторые действия по истечении заданных интервалов времени. Чаще всего для работы с таймером разработчик применяет событие `Tick`. Данное событие инициируется только в том случае, если свойство `Enabled` имеет значение `True`. Если нужно остановить таймер, то достаточно присвоить данному свойству значение `False`.

Интервал отсчета времени задается свойством `Interval`, а его значение указывает используемый промежуток времени в миллисекундах. Если рабочий интервал таймера должен составлять 3 с, то надо установить значение 3000. Этот элемент управления уже применялся при работе с объектом `ProgressBar`.

Элементы управления OpenFileDialog и SaveFileDialog

Практически в каждом приложении пользователь должен иметь возможность сохранить файл или открыть его. Разработчикам регулярно приходится реализовывать подобную функциональность в своих программах. При желании можно самому придумать и разработать интерфейс для подобной задачи. Но можно воспользоваться и стандартными диалоговыми окнами открытия и сохранения файла. Именно для этого применяются элементы управления `OpenFileDialog` и `SaveFileDialog`. К сожалению, в версии .NET Compact Framework возможности данных элементов управления серьезно урезаны. Разработчик может манипулировать файлами только в пределах папки «My Documents» и вложенных папок следующего уровня. Поэтому папка `My Documents\Programming\Sample` будет уже недоступна.

Рассматриваемые элементы управления размещаются в нижней части дизайнера формы рядом с элементом `MainMenu`. При работе с данными элементами, прежде всего, надо задать свойство `Filter`, которое ограничивает список доступных файлов, фильтруя их по расширению. Свойство `InitalDirectory` содержит имя папки, в которой по умолчанию располагаются файлы. Если это свойство оставить пустым, то обзор файлов начнется с самой папки «My Documents».

Основным методом для этих элементов является `ShowDialog`. После его вызова на экране отображается модальное окно, в котором пользователь должен нажать кнопку `OK` или `Cancel`. При этом метод ShowDialog возвращает значения `DialogResult.OK` и `DialogResult.Cancel` соответственно. Если получено значение `DialogResult.OK`, то пользователь нажал кнопку `OK` и в свойстве `Filename` содержится полный путь

к выбранному файлу. Пример работы с элементами OpenFileDialog и SaveFileDialog приведен в листинге 3.11.

Листинг 3.11.

```
private void butOpen_Click(object sender, EventArgs e)
{
     ofd.Filter = "DLL|*.dll|Картинки|*.jpg":
     ofd.InitialDirectory = "\\My Documents\\Templates";
     if (DialogResult.OK == ofd.ShowDialog()){
          statusBarl.Text = ofd.FileName;
     }
     else{
          statusBarl.Text = "Вы нажали на кнопку Отмена!";
     }
}
```

Элемент управления ImageList

Элемент управления ImageList уже рассматривался при знакомстве с элементом ToolBar. Элемент ImageList используется для хранения коллекций растровых изображений. Как и многие другие элементы, список рисунков не отображается во время выполнения программы, а используется как контейнер, из которого по мере необходимости извлекаются хранимые изображения. Как правило, данный элемент используется совместно с такими элементами управления, как ListView, TreeView и ToolBar.

Изображения можно добавлять в элемент управления во время работы приложения. Для этого используется метод Add, который входит в состав члена класса Images. Сами картинки могут располагаться как в отдельных файлах, так и в ресурсах приложения. В листинге 3.12 показано, как можно добавить картинку из ресурсов в ImageList, а затем отобразить ее в элементе интерфейса PictureBox.

Листинг 3.12.

```
Bitmap image = new Bitmap(Assembly.GetExecutingAssembly();
GetManifestResourceStream(@"ImageList.home.gif"));
imgList.Images.Add(image);
picTest.Image = imgList.Images[0];
```

Изображение добавляется в начало списка, и его порядковый номер будет равен нулю. Если в ImageList уже было одно изображение, то новая картинка будет иметь порядковый номер, равный единице. Это иллюстрируется листингом 3.13.

Листинг 3.13.

```
private void butFromlmageLlst ClickCobject sender,
     EventArgs e)
{
     picTest.Image = imgList.Images[1];
```

```
}
```

Все картинки, находящиеся в ImageList, имеют одинаковый размер. По умолчанию используется размер 16×16 пикселов. Разработчик может изменить размеры изображений, используя свойство ImageSize. Если менять отображаемые картинки при помощи таймера, то можно даже создать небольшую мультипликацию. Для этого достаточно список рисунков заполнить набором изображений, а затем поочередно отображать их в графическом поле.

Элемент управления PictureBox

Элемент управления PictureBox используется для отображения графики. Данный элемент имеет ограниченную функциональность и не позволяет растягивать картинку в соответствии с размерами графического поля. В листинге 3.14 приведен фрагмент кода, который позволяет загрузить изображение из графического файла.

Листинг 3.14.

```
private void butFromFile_Click(object sender, EventArgs e)
{
     picTest.Image = new Bitmap(@"\Windows\banner.gif");
}
```

Если использовать этот способ для добавления картинки, то нужно добавить изображение в проект и для свойства BuildAction в окне свойств Properties задать значение Content. В процессе подготовки приложения к инсталляции изображение будет рассматриваться как часть программы. В рассмотренном примере использовалась готовая картинка, которая находится в папке Windows.

Также можно загрузить изображение из ресурсов приложения. В этом случае надо добавить картинку в проект и для свойства BuildAction задать значение «Embedded Resource». Тогда не придется специально включать изображения в состав инсталлятора. В листинге 3.15 приведен пример, иллюстрирующий добавление изображения из ресурсов.

Листинг 3.15.

```
private void butRes_Click(object sender, EventArgs e)
{
     // Загружаем из ресурсов
     picTest.Image = new
     Bitmap(Assembly.GetExecutingAssembly().
     GetManifestResourceStream
          ("PictureBox_CS.kristina.jpg"));
}
```

Элемент ImageList имеет свойство ImageSize, которое задает размеры хранимых изображений. Перед загрузкой картинки в графиче-

ское поле можно установить требуемые размеры изображения с помощью данного свойства, как показано в листинге 3.16.

Листинг 3.16.

```
private void butImgList_Click(object
sender, EventArgs e)
{
     // изменяем размеры картинки
     imageList1.ImageSize =
          new System.Drawing.Size(160,
     120);
     // загружаем картинку с измененными
     размерами
     picTest.Image =
     imageList1.Images[0];
}
```

Рис. 3.18. Пример работы с элементом PictureBox

На рис. 3.18 показан внешний вид приложения, в котором для работы с изображениями применяются все три описанных варианта.

Элемент управления TabControl

Элемент управления `TabControl` очень удобен при создании интерфейсов для устройств с малыми размерами экрана, так как он позволяет создавать многостраничные диалоговые окна. Вкладки, реализуемые этим элементом, имеют ярлычки, на которых отображаются заголовки страниц. И пользователь может легко переключаться между страничками, просто щелкая по этим ярлычкам.

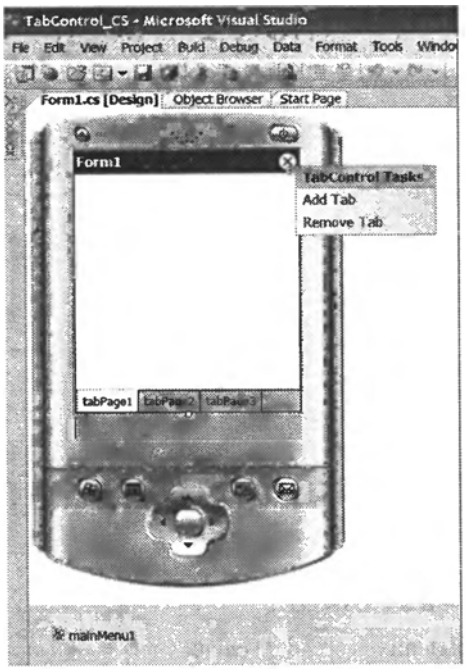

Рис. 3.19. Пример работы с элементом TabControl

В устройствах Pocket PC вкладки располагаются в нижней части окна. Следует обратить внимание на то, что элемент `TabControl` всегда располагается в верхнем левом углу контейнера. Например, если поместить `TabControl` на форму, то он появится в ее верхнем левом углу. Если же нужно изменить расположение этого элемента, то надо поместить его на панель, которая является контейнером. При перемещении панели будет перемещаться и `TabControl`.

Элемент `TabControl` следует расположить на форме. У него по умолчанию будут созданы вкладки `tabPage1`

и tabPage2. Если нужно добавить новую вкладку, то следует щелкнуть на маленькой стрелке в верхней части элемента TabControl и выбрать пункт меню «Add Tab» (рис. 3.19).

В результате у элемента TabControl появится новая закладка, которую можно настроить в соответствии с потребностями разработчика.

Программист также может воспользоваться услугами редактора «TabPage Collection Editor» для добавления новых закладок. В этом случае надо выбрать элемент TabControl в дизайнере формы, найти свойство TabPages и нажать кнопку редактирования этого свойства. В результате будет открыт редактор закладок. Для управления закладками можно также выделить TabControl, щелкнуть на нем правой кнопкой мыши и выбрать пункты контекстного меню «Add Tab» или «Remove Tab».

Для определения текущей вкладки используется свойство SelectedIndex. При изменении данного свойства инициируется событие SelectedIndexChanged, что иллюстрирует код, приведенный в листинге 3.17.

Листинг 3.17.

```
private void tabControl1_SelectedIndexChanged(object sender,
EventArgs e)
{
switch (tabControl1.SelectedIndex)
{ case 0:
     MessageBox.Show("Вы выбрали первую вкладку");
     break;
  case 1:
     MessageBox.Show("Вы выбрали вторую вкладку");
     break;
  case 2:
     MessageBox.Show("Вы выбрали третью вкладку");
     break;
}
```

Элемент управления InputPanel

Элемент управления InputPanel позволяет вводить текстовую информацию при помощи виртуальной клавиатуры или панели распознавания знаков Soft Input Panel (SIP). Так как в полной версии .NET Framework данного элемента нет, то стоит рассмотреть его несколько подробнее.

Как правило, в карманных компьютерах нет клавиатуры, поэтому для ввода данных используется виртуальная клавиатура. В обычном состоянии она неактивна и находится в свернутом состоянии. Чтобы ее активировать, нужно щелкнуть стилусом по значку клавиатуры в нижнем правом углу экрана, где располагается меню или панель инструмен-

тов ToolBar. Тогда виртуальная клавиатура появится на экране, и пользователь сможет вводить текст.

Разработчик может программно управлять состоянием клавиатуры. Например, с помощью элемента `InputPanel` можно узнать текущее состояние SIP, возможность ее отображения и деактивации. Свойств и методов у элемента `InputPanel` не так уж много, но наиболее часто используемые члены класса приведены в следующем списке.

Свойства `Bounds` и `VisibleDesktop` доступны только для чтения и определяют прямоугольники, по которым можно судить о положении SIP и размерах клиентской области, не занятой SIP. Свойство `VisibleDesktop` определяет прямоугольник, которым ограничена область экрана, не закрытая SIP. Когда виртуальная клавиатура отображается, то условный прямоугольник поднимается от полоски навигации над окном SIP. Когда SIP скрыт, то прямоугольник занимает все пространство экрана.

Свойство `Bounds` описывает размеры и позицию виртуальной клавиатуры, когда она отображается на экране. Причем размеры этого прямоугольника не меняются, даже если виртуальная клавиатура скрыта. Свойство `Enabled` имеет значение `True`, если виртуальная клавиатура отображается на экране. Значение свойства можно задавать программно, скрывая или отображая клавиатуру.

Если в приложении необходимо использовать элемент `InputPanel`, нужно следить, чтобы он при активации не загородил элементы управления на форме, иначе пользователь просто не сможет ввести данные.

Можно размещать текстовые поля на форме как можно выше, чтобы элемент `InputPanel` не мешал вводить данные. Но помимо этого можно отслеживать состояние виртуальной клавиатуры и при ее отображении передвигать вверх поля для ввода информации. В этом случае чаще всего приходится применять полосы прокрутки. Можно проиллюстрировать такое поведение примером, в котором элемент `InputPanel` будет активироваться, если текстовое поле получит фокус. А когда `TextBox` потеряет фокус, то клавиатура должна исчезнуть. Эта функциональность реализуется кодом, приведенным в листинге 3.18.

Листинг 3.18.

```
private void txtTest_GotFocus (object sender, EventArgs e)
{
     // Когда пользователь выбирает текстовое поле.
     // то автоматически активируем SIP
     inputPanel1.Enabled = true;
}
private void txtTest_LostFocus (object sender, EventArgs e)
```

```
{
     // При потере фокуса сворачиваем SIP
     inputPanel1.Enabled = false;
}
```

В этом примере на форму добавлена кнопку, так как иначе текстовое поле будет автоматически получать фокус при загрузке формы и приложение не будет работать так, как это задумывалось изначально. В данном случае именно кнопка будет получать фокус ввода при загрузке формы. Но для этого надо присвоить свойству кнопки `TabIndex` нулевое значение.

Теперь если стилусом активировать текстовое поле, то автоматически будет отображена виртуальная клавиатура. Если на появившейся клавиатуре нажать клавишу Tab или щелкнуть на кнопке с надписью «Просто кнопка», то панель ввода автоматически свернется.

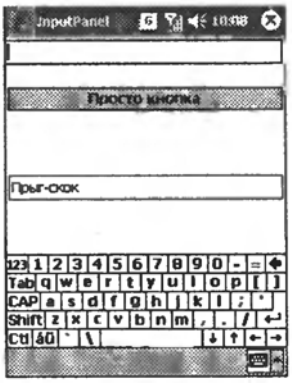

Рис. 3.20. Окно тестирования приложений

Элемент `InputPanel` поддерживает событие `EnabledChanged`, которое возникает при активации и деактивации панели ввода. С его помощью можно расширить функциональность примера. Следует добавить еще одно текстовое поле в нижнюю часть формы. Сейчас нужно отследить событие `EnabledChanged` и отреагировать на него должным образом. При активации панели ввода текстовое поле должно сдвинуться вверх, чтобы пользователь мог ввести свои данные. Пример обработки этого события приведен в листинге 3.19.

Листинг 3.19

```
private void inputPanel1_EnabledChanged(object sender,
          EventArgs e)
{
// Отслеживаем состояние панели ввода
// Свойство Bounds возвращает размеры и позицию SIP
    if(inputPanel1.Enabled = true)
     this.txtJump.Top = 200 - inputPanel1.Bounds.Height;
    else
     this.txtJump.Top = 200;
}
```

На рис. 3.20 показан внешний вид окна тестового приложения.

Элемент управления DataGrid

Элемент управления `DataGrid` позволяет отображать данные в виде таблицы, как это сделано в электронной таблице MS Excel. Как

и многие другие элементы управления, он имеет урезанные возможности по сравнению с полной версией .NET Framework. Например, отключена поддержка свойства `DataMember`. Элемент управления `DataGrid` может быть связан с источниками данных при помощи свойства `DataSource`. Рассмотрим простейший пример работы с данным элементом. Прежде всего, потребуется создать XML-файл, содержащий некоторые данные. Для примера был использован файл `artists.xml`, в котором содержится информация о некоторых известных артистов шоу-бизнеса. Файл содержит записи о фамилии, дате рождения и адресе проживания. Созданный файл нужно добавить в проект, расположить на форме элемент `DataGrid` и присвоить ему имя `grdArtists`. В листинге 3.20 приведен код обработчика события `Form1_Load`.

Листинг 3.20.

```
private void Form1 Load(object sender, EventArgs e)
{
   Cursor.Current = Cursors.WaitCursor;
   try
   {
   // Загружаем данные DataSet
      DataSet ds = new DataSet();
      ds.ReadXml(@"\Program Files\DataGrid\artists.xml");
      grdArtists.DataSource = ds.Tables[0];
   }
   catch (Exception)
   {
      MessageBox.Show
   ("Не могу загрузить данные в DataGrid! ", this.Text);
      // Устанавливаем стили
      DataGridTableStyle ts = new DataGridTableStyle();
      ts.MappingName = "Order";
      DataGridColumnStyle artistDate =
          new DataGridTextBoxColumn();
      artistDate.MappingName = "BirthDate";
      artistDate.HeaderText = "Дата рождения";
      ts.GridColumnStyles.Add(artistDate);
      DataGridColumnStyle artistName =
          new DataGridTextBoxColumn();
      artistName.MappingName = "ArtistName";
      artistName.HeaderText = "Артист";
      artistName.Width = this.Width - artistDate.Width - 35;
      ts.GridColumnStyles.Add(artistName);
      grdArti sts.Tab!eStyles.Add(ts);
      Cursor.Current = Cursors.Default;
   }
```

В данном примере из XML-файла извлекаются данные, относящиеся к фамилии артиста и дате его рождения.

Остальная информация игнорируется. При загрузке приложения в элемент DataGrid записываются требуемые данные. На рис. 3.21 показан внешний вид приложения.

Рис. 3.21. Пример работы с элементом DataGrid

Элемент управления MonthCalendar

Данный элемент позволяет легко выбрать необходимую дату. Для создания тестового приложения на форме надо разместить элементы `MonthCalendar` и `Label`. Метка должна получить имя `lblSelectDate`, а для свойства `Text` нужно задать значение Выбранная дата. Затем следует дважды щелкнуть на элементе `monthCalendar1`, чтобы задать код обработчика события `DateChanged`. Этот код приведен в листинге 3.21

Листинг 3.21.

```
private void monthCalendar1_DateChanged(object sender,
     DateRangeEventArgs e)
{
     lblSelectDate.Text = "Выбранная дата: " +
          monthCalendar1.SelectionStart.ToShortDateString();
}
```

Теперь можно запустить приложение и выбрать любую дату из месячного календаря. Выбранная дата будет автоматически отображаться в надписи `lblSelectDate`, как показано на рис. 3.32.

Элемент управления DocumentList

Элемент управления `DocumentList` может заменить такие элементы, как `SaveFileDialog` и `OpenFileDialog`, так как имеет все необходимые средства для работы с файлами. Помимо этого элемент `DocumentList` имеет дополнительные возможности, которые позволяют очень просто реализовать основные задачи манипулирования файлами, к которым относятся копирование, удаление, переименование и перемещение файлов. С помощью этого элемента также можно сортировать файлы по имени, дате создания, размеру. Кроме того, существует даже возможность посылать файлы по электронной почте или передавать на другое устройство при помощи инфракрасной связи. Элемент `DocumentList` работает с файлами в пределах папки «My Documents», включая подпапки.

Для разработки тестового приложения сначала потребуется создать новый проект, а затем переместить на форму элемент `DocumentList`.

Для свойства Name надо задать значение DocListFile, свойство Dock должно получить значение Top, свойству Heigth присваивается значение 160, а для свойства SelectedDirectory задается значение «My Documents».

Также на форме надо разместить элементы ComboBox и StatusBar. Элементу ComboBox надо присвоить имя cboFileType. Затем следует выбрать свойство Items и открыть окно редактора «String Collection Editor». Для списка надо задать значения BMP и WAV. Затем нужно дважды щелкнуть на элементе ComboBox, чтобы задать код обработчика события SelectedIndexChanged. Код обработчика приведен в листинге 3.22.

Рис. 3.22. Выбор файла при помощи элемента DocumentList

Листинг 3.22.

```
private void cboFileType_SelectedIndexChanged(object sender,
     EventArgs e)
{
   if(cboFileType.Text = "BMP")
   {
      docListFile.Filter = "Рисунки (*.bmp)|*.bmp";
      docListFile.SelectedDi rectory = "My Pictures";
   }
   else
   {
      docListFile.Filter = "Звуки (*.wav)|*.wav";
      docListFile.SelectedDirectory = "My Music";
   }
}
```

Данный код динамически меняет значение свойства Filter элемента DocumentList для отображения файлов определенного типа. Также меняется папка просмотра файлов. Если пользователь выберет расширение .BMP, то следует выбрать папку «My Pictures», специально предназначенную для хранения картинок. При выборе типа файлов .WAV выбирается папка «My Music». Теперь следует

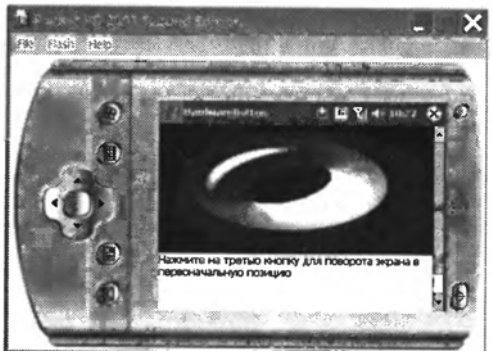

Рис. 3.23. Поворот экрана при помощи аппаратных кнопок

дважды щелкнуть на элементе DocumentList, чтобы создать обработчик события DocumentActivated. Соответствующий код приведен в листинге 3.23.

Листинг 3.23.

```
private void docListFile_DocumentActivated(object sender,
     DocumentListEventArgs e)
{
     statusBar1.Text = e.Path;
     // работа с выбранным файлом
}
```

Перед началом тестирования стоит скопировать несколько соответствующих файлов в папки «My Pictures» и «My Music». После запуска программы нужно перейти в поле со списком и выбрать тип файлов. После этого будет активирован элемент DocumentList с выбранной папкой. Из списка документов можно будет выбрать конкретный файл.

Следует обратить внимание на то, что выбранный файл имеет контекстное меню, при помощи которого можно выполнять базовые операции с файлом (рис. 3.22). Путь к выбранному файлу отображается в строке состояния.

Элемент управления HardwareButton

На карманных компьютерах кроме клавиш навигации присутствуют также дополнительные кнопки, при помощи которых активируются часто запускаемые приложения. Как правило, в состав программного обеспечения КПК входит утилита, с помощью которой можно назначить каждой из этих кнопок определенные команды. Но можно представить ситуацию, когда для создаваемой игры нужно, чтобы управление осуществлялось с помощью этих кнопок. Тогда необходимо переопределить на время поведение кнопок в вашем приложении. И сделать это можно с помощью элемента HardwareButton. Для пояснения на примере использования этого элемента, прежде всего, нужно создать новый проект и поместить на панели «Component tray» два элемента HardwareButton с именами hrdLeftRotate и hrdRightRotate. Для каждой переопределяемой кнопки необходимо создать свой экземпляр элемента HardwareButton.

В рассматриваемом примере будут переопределяться вторая и третья кнопки. Также на форме надо поместить графическое поле PictureBox. В него надо загрузить любое изображение и растянуть картинку таким образом, чтобы она заняла верхнюю половину экрана. Изображение надо пристыковать к верхней части формы. Для этого свойству Dock присваивается значение Top. Также на форме надо поместить надпись Label, при помощи которой будут отображаться подсказки. Надпись

следует пристыковать к нижней части формы. Для этого свойству Dock присваивается значение Bottom. У обоих добавленных элементов HardwareButton нужно отыскать свойство AssociatedControl и задать значение Form1. Также надо изменить значения свойств HardwareKey. Для первого элемента применяется значение ApplicationKey2, что соответствует второй кнопке. Для второго элемента задается значение ApplicationKey3, что соответствует третьей кнопке под экраном. Теперь, когда все необходимые свойства установлены, нужно написать код для события Form1_KeyUp. Код приведен в листинге 3.24.

Листинг 3.24.

```
private void Form1_Load( object sender, EventArgs e)
{
   Label1.Text = "Нажмите вторую кнопку для " +
     "поворота экрана на 90 градусов";
}
private void Form1_KeyUp( object sender, KeyEventArgs e)
{
   switch ((HardwareKeys)e.KeyCode)
   {
   case HardwareKeys.ApplicationKey2:
      if(SystemSettings.ScreenOrientation ==
            ScreenOrientation.Angle0)
      {
         SystemSettings.ScreenOrientation =
               ScreenOrientation.Angle90:
         label1.Text = "Нажмите третью кнопку для " +
            "поворота экрана в первоначальную позицию";
      }
   break;
   case HardwareKeys.ApplicationKey3:
      if(SystemSettings.ScreenOrientation ==
            ScreenOrientation.Angle90)
      {
         SystemSettings.ScreenOrientation =
               ScreenOrientation.Angle0;
          Label1.Text = "Нажните вторую кнопку для " +
            "поворота экрана на 90 градусов";
      }
      break;
      default:
      break;
   }
}
```

Запустите программу и попытайтесь нажимать по очереди на вторую и третью кнопки под экраном карманного компьютера. Результат показан на рис. 3.23.

3.2.3. Работа со стилусом и клавиатурой

Стилус

Взаимодействие с программой пользователь осуществляет с помощью стилуса или аппаратных кнопок на самом устройстве. На мобильном устройстве роль мыши с успехом выполняет стилус. Однако, стоит заметить, что у стилуса нет возможности эмулировать нажатие правой кнопки мыши. Когда пользователь применяет стилус, то генерируются события `MouseDown`, `MouseMove`, `MouseUp` и `Click`. Первые три события могут сообщить о позиции курсора, как и события из настольной версии Windows.

Курсоры

Так как пользователь при работе использует стилус, то Windows Mobile не отображает на экране устройства стандартную стрелку курсора. Предполагается, что пользователь может самостоятельно попасть острым концом стилуса в маленькую кнопку или другой элемент. Но у мобильных систем курсоры, все таки, есть. Первый из них является аналогом песочных часов в настольной версии Windows и выглядит как анимированный круг с разноцветными секторами. Второй курсор можно увидеть при вызове контекстного меню. Он выглядит как множество маленьких красных кружков, которые постепенно появляются вдоль воображаемой окружности.

Песочные часы

При выполнении длительных ресурсоемких операций нужно по казать пользователю, что устройство работает, а не зависло. Лучше всего вывести на экран устройства курсор ожидания. В карманных компьютерах в качестве такого курсора используются не песочные часы, как в настольных компьютерах, а анимированный разноцветный круг. Установить данный тип курсора в приложении очень просто, что иллюстрирует фрагмент кода, приведенный в листинге 3.25.

Рис. 3.24. Отображение курсора ожидания

Листинг 3.25.

```
// Устанавливаем курсор ожидания
   Cursor.Current = Cursors.WaitCursor;
// Возвращаем курсор по умолчанию
   Cursor.Current = Cursors.Default;
```

На рис. 3.24 показано вид приложения с соответствующим курсором.

Обработка события «Tap-and-Hold»

Так как в карманных компьютерах не используется правая кнопка мыши, то для вызова контекстного меню используется операция «Tap-and-Hold». Пользователь касается стилусом экрана и некоторое время удерживает нажатие. Если элемент, на поверхности которого находится стилус, связан с элементом `ContexMenu`, то на экране появится контекстное меню. Если нужно создать собственный обработчик события Tap-and-Hold, то нужно добавить в проект таймер и написать код для обработки событий `Mouse_Down`, `MouseUp` и `timer1_Tick`. Для таймера следует задать интервал, необходимый для инициирования события. Сам код приведен в листинге 3.26.

Листинг 3.26.

```
private void Form1_MouseDown(object sender,MouseEventArgs e)
{
   // включаем таймер
   timer1.Enabled = true;
}
private void Form1_MouseUp(object sender, MouseEventArgs e)
{
   timer1.Enabled = false;
   label1.Text = "";
}
private void timer1_Tick(object
     sender, EventArgs e)
{
   label1.Text = "Вы нажали на экран";
}
```

Клавиатура

На большинстве карманных компьютеров нет стандартной клавиатуры, поэтому ввод текста осуществляется с помощью виртуальной клавиатуры SIP. В Visual Studio клавиатура SIP представлена элементом `InputPanel`. Но в последнее время стали появляться устройства с настоящей встроенной клавиатурой. Как правило, эти устройства имеют квадратный экран. Среда разработки поддерживает эмуляторы подобных моделей (рис. 3.25). Эти эмуляторы в своем названии содержат слово «Square». Кроме того, на устройствах имеются клавиши навигации, клавиша Enter и кнопки запуска определенных приложений. Все эти клавиши инициируют стандартные события.

Рис. 3.25. Эмулятор устройства с клавиатурой

Клавиши навигации

Если в процессе создания приложения в режиме работы с формой щелкнуть мышью на любой из кнопок навигации, то среда разработки сгенерирует код для этих кнопок в событии `Form1_KeyDown`. В листинге 3.27 приведен пример обработчика этого события.

Листинг 3.27.

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
   if (e.KeyCode == System.Windows.Forms.Keys.Up)
   {
      label1.Text = "Клавиша Вверх";
   } if (e.KeyCode == System.Windows.Forms.Keys.Down)
   {
      label1.Text = "Клавиша Вниз";
   } if (e.KeyCode == System.Windows.Forms.Keys.Left)
   {
      label1.Text = "Клавиша Влево";
   } if (e.KeyCode == System.Windows.Forms.Keys.Right)
   {
      label1.Text = "Клавиша Вправо";
   } if (e.KeyCode == System.Windows.Forms.Keys.Enter)
   {
      label1.Text = "Клавиша Enter";
   }
}
```

Как видно из данного примера, приложение определяет нажатую клавишу при помощи перечисления `System .Windows.Forms.Keys`. Если открыть виртуальную клавиатуру и нажать на клавиши со стрелками, то можно убедиться, что они тоже инициируют событие `Form1_KeyDown` (рис. 3.26). Если протестировать пример на устройстве с настоящей клавиатурой, то можно заметить, что приложение правильно обрабатывает нажатие на встроенные клавиши со стрелками.

Рис. 3.26. Обработка нажатий клавиш навигации

3.2.4. Разработка мобильных приложений

Активация и деактивация формы

Модель выполнения программ на карманном компьютере отличается от поведения программ, работающих на обычном персональном компьютере. Например, на мобильных компьютерах используется один

экземпляр запущенной программы. Аналогом подобного поведения на настольных компьютерах является почтовая программа Outlook Express, которая всегда запускается в одном экземпляре. При попытке запуска программы она просто активируется (если уже была запущена). При этом вторая копия программы не запускается.

При создании приложения для КПК разработчику не придется прилагать никаких усилий для реализации подобного поведения. Среда выполнения .NET Compact Framework сама позаботится о том, чтобы запускался только один экземпляр программы. Следует помнить, что пользователь не должен сам закрывать программу. При запуске новой программы ее окно просто загораживает предыдущую программу.

Учитывая подобное поведение, нужно писать программы, которые не занимают много ресурсов системы. Однажды запущенное приложение может находиться в памяти несколько дней, пока пользователь не перезагрузит компьютер или не закроет программу самостоятельно. Деактивированная программа закроется автоматически, если система обнаружит уменьшение свободной памяти при разрядке батареи. Но, тем не менее, иногда надо проследить, чтобы при закрытии программа освободила ресурсы, которые она использовала. Бывают ситуации, когда приложение поддерживает соединение с базой данных или осуществляет связь с COM-портами. В этом случае система может не освободить занимаемые программой ресурсы. Для отслеживания состояния формы используются события `Form.Deactivate` и `Form.Activated`. В листинге 3.28 приведен пример работы с этими событиями.

Листинг 3.28.

```
private void Form1_Activated(object sender, EventArgs e)
{
// Здесь ваш код для восстановления связей с портами и т. д.
   lblInfo.Text = "Приложение активировано";
}
private void Form1_Deactivate( object sender, EventArgs e)
{
// Здесь ваш код для освобождения ресурсов
   lblInfo.Text = "Приложение деактивировано";
}
```

Так как приложение в неактивном состоянии может быть закрыто системой, то важно блокировать возможную потерю данных. Для этого нужно использовать событие `Deactivate`.

Создание собственных диалоговых окон

Сложные приложения часто используют несколько форм. Например, во многих программах имеется диалоговое окно о программе, в ко-

тором отображаются информация о программе, номер версии, сведения об авторе и логотип компании.

Для создания подобных форм хорошо подойдет собственное диалоговое окно. Чтобы отобразить такое окно, используется метод `ShowDialog`. Этот метод делает недоступным родительскую форму, пока диалоговое окно находится на экране. Диалоговое окно может возвращать результат вызова метода `ShowDialog` не только себе, но и родительскому окну.

Например, для создания специального окна авторизации пользователя для доступа к программе, в состав проекта нужно включить новую форму, которая будет реализована как диалоговое окно проверки имени пользователя `LogonForm`. Это будет маленькое окно, в котором надо разместить текстовое поле и две кнопки. Затем надо задать значения свойств `FormBorderStyle`, `Size` и `Location`. При загрузке основной формы и обработке события `Load` создается новый экземпляр объекта `LogonForm` и вызывается как диалоговое окно. Данное окно может вернуть значения `DialogResult.OK`, если пользователь ввел имя, или `DialogResult.Cancel`, если он просто закрыл форму. Если было введено правильное имя, то главная форма продолжает свою работу. В противном случае приложение следует закрыть. Соответствующий код приведен в листинге 3.29.

Листинг 3.29.

```
private void Form1_Load(object sender, EventArgs e)
{
   LogonForm LogonFrm = new LogonForm();
   if(LogonFrm.ShowDialog() == DialogResult.Cancel)
   {
      LogonFrm.Dispose();
      this.Close();
   }
   else
   {
      this.Text += " - " + LogonFrm.txtCheck.Text;
      LogonFrm.Dispose();
   }
}
```

После того как форма авторизации будет отображена на экране, нужно обработать события `Click` для нажатия кнопки проверки введенного имени пользователя или кнопки отмены. Первая кнопка проверяет правильность ввода имени. Если проверка завершилась успешно, то возвращается значение `DialogResult.OK`. Это иллюстрирует код, приведенный в листинге 3.30.

Листинг 3.30.

```
private void butOK Click(object sender, EventArgs e)
{
   if(txtCheck.Text == "Alex")
   {
      this.DialogResul t = DialogResult.OK;
   }
   else
   {
      MessageBox.Show("В доступе отказано. " +
         "Попробуйте еще раз ", "Вход в программу");
   }
}
```

Если пользователь не знает имени для доступа к программе, то ему придется нажать кнопку «Отмена». В этом случае обработчик события `butCancel_Click`, код которого приведен в листинге 3.31, возвращает значение `DialogResult.Cancel` в главную форму, которая закрывает приложение.

Листинг 3.31.

```
private void butCancel_Click(object sender,
          System.EventArgs e)
{   this.DialogResult = DialogResult.Cancel;}
```

Распространение мобильных приложений

Программы для настольных компьютеров распространять довольно просто. Нужно лишь создать специальный проект для создания установочного пакета, который сгенерирует специальный файл установки «Microsoft Installer» (MSI). К сожалению, для мобильных устройств, процесс создания установочных файлов немного отличается. В процессе распространения программы участвуют три составляющие: настольный компьютер, программа синхронизации «Microsoft ActiveSync» и программа `wceload.exe` для извлечения файлов из cab-файлов.

Для пользователя процесс установки программы не сильно отличается от привычной схемы. Сначала он скачивает программу или находит ее на компакт-диске. Затем запускает установочный msi-файл. Программа Microsoft Installer с помощью специального мастера установки помогает пользователю установить программу с нужными настройками. После этого программа считается установленной, и пользователь может запускать ее.

Создание cab-файла

Прежде чем установочный пакет попадет в руки пользователя, необходимо поработать над его созданием. Устройства под управлением Windows Mobile не могут напрямую работать с файлами *.msi. Вместо этого исполь-

зуются файлы с расширением .cab (кабинетные файлы). Таким образом, задача программиста заключается в том, чтобы составить список команд для программы синхронизации ActiveSync, которые позволят скопировать cab-файлы на устройство с учетом необходимых установок. Для создания удобного установочного пакета с интуитивно понятным интерфейсом вам необходимо выполнить несложную последовательность действий.

1. Создать cab-файл для устройства.
2. Добавить в cab-файл дополнительные файлы, используемые программой, например изображения или файлы с данными.
3. Добавить в cab-файл инструкции для записи в реестр.
4. Зарегистрировать cab-файл с помощью ActiveSync, чтобы пользователь мог установить приложение с настольного компьютера.
5. Написать код для различных дополнительных возможностей, которые будут использоваться установочным пакетом во время установки или деинсталляции.
6. Упаковать все необходимые файлы в один специальный файл установки с расширением *.msi.

Известно, что кабинетный файл является специальным файлом упаковки, с помощью которого можно сжимать файлы, что приводит к уменьшению их размеров. В этом файле также могут содержаться инструкции для внесения изменений в реестр системы. За обработку cab-файлов на устройстве отвечает утилита `wceload.exe`, входящая в состав Windows Mobile.

Создание проекта распространения разработанной программы

Для создания установочного пакета нужно запустить уже существующий проект, который планируется подготовить для распространения, например, `MyMobileApp`.

А затем нужно выполнить команду меню File->Add->New Project. В открывшемся диалоговом окне надо перейти в раздел Other Project Types, выбрать тип «Smart Device Cab Project» и задать имя нового проекта, например, `MyMobileAppCab` (рис. 3.27).

В окне свойств надо задать значения свойств `Manufacturer` и `ProductName`. Другие свойства позволяют задать минимальные и максимальные требования к операционным системам, в которых может быть запущена ваша программа. Затем надо запустить редактор «File System Editor», нажав соответствующую кнопку в окне свойств. Нужно выбрать пункт «Application Folder» и в контекстном меню выбрать пункт Add->Project Output (рис. 3.28).

В результате этого будет открыто диалоговое окно «Add Project Output Group».

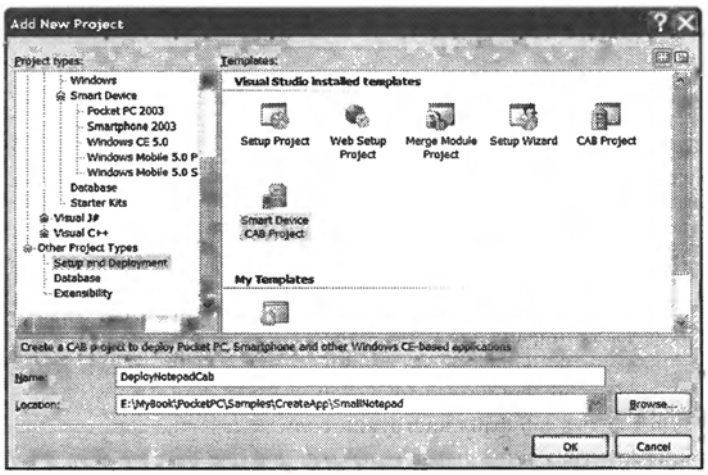

Рис. 3.27. Выбор нового проекта для распространения приложения

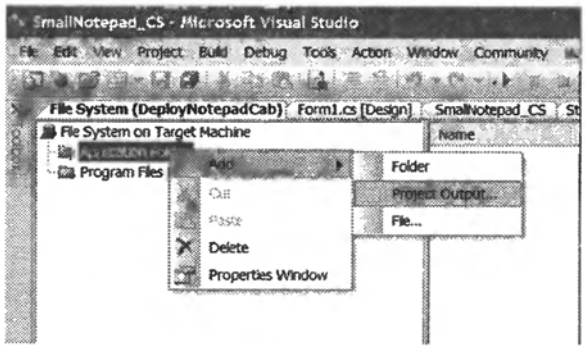

Рис. 3.28. Выбор параметров проекта

С помощью данного окна можно выбрать различные типы файлов, необходимые для программы, такие как файлы документации или, например, локализированные ресурсы. Нужно выбрать пункт «Primary Output» и нажать кнопке ОК.

В правой части окна следует щелкнуть правой кнопкой мыши на единственном пункте «Primary output from MyMobileApp» и в контекстном меню выбрать пункт «Create Shortcut to Primary output from MyMobileApp» (рис. 3.29).

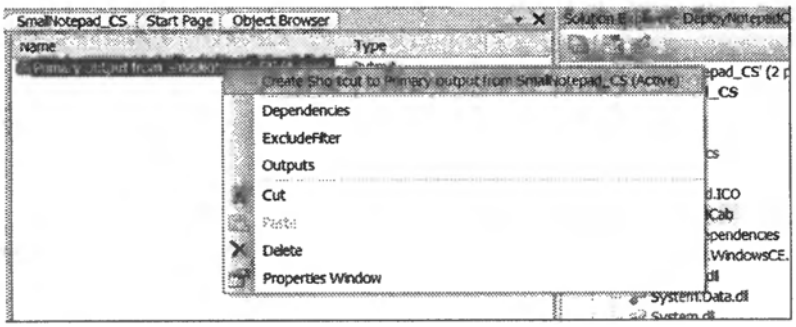

Рис. 3.29. Создание пиктограммы приложения

Это позволит включить пиктограмму в список файлов для распространения. Созданный ярлык надо переместить мышью в папку «Program Files Folder». Теперь можно приступать к созданию установочного файла. В меню надо выполнить пункт Build->Build MyMobileAppCab. После этого среда разработки создаст специальный файл с расширением .CAB. При помощи файлового менеджера его нужно найти и запомнить его расположение.

После этого нужно установить созданный файл на эмуляторе. Для этого выполняется команда меню Tools->Device Emulator Manager. В диалоговом окне надо выбрать эмулятор. В этом же окне следует выполнить команду меню Actions-> Connect.

При этом выбранный эмулятор будет активирован. В окне эмулятора надо выполнить команду меню File->Configure. После этого откроется окно настроек эмулятора, в котором следует перейти в раздел «Shared Folder». В этом разделе надо выбрать папку, в которой находится созданный cab-файл (рис. 3.30).

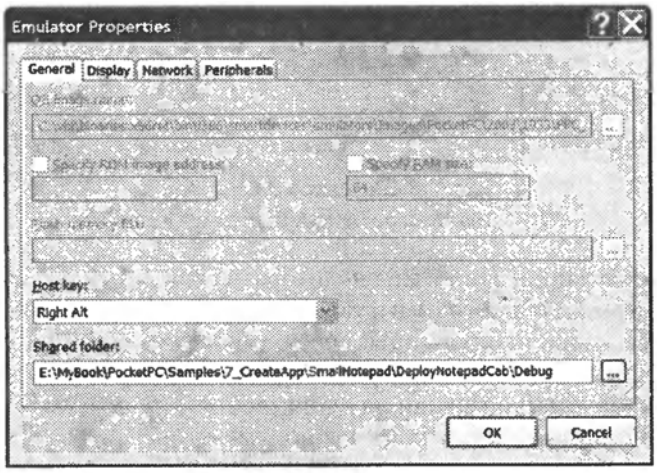

Рис. 3.30. Активация и настройка эмулятора

Эмулятор будет считать, что данная папка является карточкой памяти. Если открыть в эмуляторе программу «File Explorer» (Start->Programs->File Explorer) и найти папку «Storage Card», то в ней можно будет увидеть ранее созданные установочные файлы.

Нужно выбрать файл `MyMobileAppCab` и запустить его. В результате начнется процесс установки программы на устройство. При установке автоматически будет создан файл деинсталляции. Он поможет корректно удалить приложение. Для этого в окне эмулятора надо выполнить команду меню Start->Settings->System->Remove Program. В списке установленных программ надо найти ранее установленное приложение, выделить его и нажать кнопку Remove.

В результате этого действия будет запущен мастер удаления программы, который корректно удалит все файлы, относящиеся к приложению. На этом изучение примера создания установочного файла можно считать законченным.

3.3. Работа с web-сервисами в мобильных приложениях

Обмен данными между приложениями играет существенную роль разработке сетевых сервисов. С помощью мобильных устройств, которые находятся у пользователя всегда под рукой, они могут в любой момент получить необходимую ему информацию или услуги. Однако обеспечение подобной гибкости требует преодоления целого ряда трудностей. Разработчикам мобильного приложения, необходимо обеспечить работу мобильных устройств в сетях разного типа, неустойчивости соединений и различных режимов связи.

Существуют множество потенциальных механизмов, способных обеспечить возможности коммуникации для мобильных приложений. Web-сервисы предлагают полезный способ организации взаимодействия приложений с Internet и интрасетями. Путем использования данных в формате XML, обычно передаваемых посредством протоколов HTTP или HTTPS, web-сервисы могут предоставить мобильному приложению возможность использовать все преимуществами существующей инфраструктуры Web сети для удовлетворения коммуникационных потребностей.

Подобно технологиям, основанным на HTML, web-сервисы создают оболочки вокруг коммерческих источников информации и предоставляют интерфейсы для доступа к ним со стороны других приложений и позволяют приложениям обмениваться информацией с использованием общего высокоуровневого стандартизированного языка, основанного на XML.

Однако нужно понимать, что платой за предоставляемые web-сервисами таких возможностей являются значительные накладные расходы. XML-формат порождает текстовые файлы большого размера и не способен обеспечить эффективный обмен данными через сеть в случае крупных XML-сообщений.

3.3.1. Краткое описание web-сервисов

Web-сервисы используют специальным образом форматированные XML-сообщения для передачи и получения запросов, связанных с предоставлением определенных услуг вычислительного характера. "Вызов" web-сервисы это направляемый одним вычислительным устройством другому запрос выполнения определенного вида обработки, за-

вершение которой обычно сопровождается возвращением результата. Web-сервисы характеризуются следующими свойствами:

- ***Использование протокола SOAP***. SOAP – это простой протокол доступа к объектам (Simple Object Access Protocol), являющийся диалектом XML. Наиболее распространенная сфера применения SOAP – вызов методов на сервере и получение ответных результатов. SOAP упаковывает имя затребованного метода вместе с параметрами в XML и доставляет эти XML-данные на сервер для обработки. Сервер получает запрос SOAP в форме XML, анализирует его для извлечения имени и параметров метода, после чего вызывает этот метод. По завершении обработки метода сервером результат SOAP возвращается клиенту в виде XML. Клиент анализирует возвращенные XML-данные, извлекает результаты и предоставляет их вызывающей программе.
- ***Использование WSDL-документов.*** WSDL – это язык описания web-сервис (Web Service Description Language). WSDL-документ – это диалект XML, предназначенный для описания служб. WSDL-документ детализирует, какими методами располагает web-сервис, какие параметры принимает каждый из методов, и что собой представляют возвращаемые методами значения. В типичных случаях WSDL-документы генерируются машиной и возвращаются по запросу от сервера web-сервисов. Инструментальные программные средства разработки загружают WSDL-документы *с* серверов web-сервисов и генерируют код на стороне клиента, позволяющий упростить вызов описанных web-сервисов на этих серверах. С каждым web-сервисом связан его собственный WSDL-документ.
- ***Возможное использование коммуникационных протоколов HTTP и HTTPS для, генерации запросов и получения ответов.*** Хотя с теоретической точки зрения web-сервисы могут выполняться с использованием самых различных коммуникационных транспортных средств, включая простой протокол электронной почты SMTP, на практике большинство запросов web-сервисам пересылаются посредством протоколов HTTP или HTTPS. Это означает, что большинство web-сервисы выполняются на Web-серверах тех же видов, которые обслуживают приложения на основе HTML. Это обстоятельство особенно полезно по той причине, что запросы HTTP и HTTPS обычно пропускаются брандмауэрами, благодаря чему доступ к web-сервисам оказывается столь же простым, что и доступ к web-страницам. Если данный Web-сервер доступен вашему мобильному приложению, то возможен и доступ к web-сервисам, выполняющимся на этом сервере.

В дополнение к этим базовым характеристикам web-сервисов разрабатываются дополнительные слои технологии, располагающиеся поверх SOAP и WSDL, которые позволят удовлетворить запросы более сложной природы. Например, такие спецификации, как WS-Reliability и WS-Security, обеспечивают потребности в надежной доставке информации и встроенных средствах безопасности. Постоянно создаются, обсуждаются, развиваются и стандартизируются и другие спецификации. В ближайшие годы технологии web-сервисов будут интенсивно расширяться и развиваться

3.3.2. Создание web-сервисов

Чтобы создать web-сервис для целей тестирования, необходимо запустить сервер Internet Information Server (IIS) (на локальном компьютере, либо на сервере, доступном для тестируемого мобильного устройства). Для инсталляции IIS потребуется также наличие соответствующих серверных расширений, обеспечивающих функционирование вашего средства разработки; в конфигурировании этих расширений на IIS вам поможет инсталлятор Visual Studio .NET.

Для создания web-сервиса необходимо выполнить следующие действия:

1. Запустите Visual Studio и создайте новый проект C# «ASP.NET Web service». Средство разработки попросит вас указать местоположение web-сервиса. При задании, например, адреса `http://localhost/WebService1`, web-сервис будет создаваться на локальном компьютере, тогда как указание адреса `http://MyWebServer/WebService1` приведет к созданию web-сервиса с именем `WebService1` на сервере с именем `MyWebServer`. В результате будет создан класс `Service1`, а на указанном в проекте web-сервере будет развернут файл `Service1.asmx`.
2. Создайте в классе `Service1` общедоступный web-метод. Приведенный в листинге 3.32 код соответствует простому методу, предоставляющему себя в качестве web-сервиса. Введите код в класс `Service1`, созданный ранее на шаге 1.
3. Чтобы развернуть и запустить на выполнение проект web-сервиса, нажмите клавишу <F5>. Для web-сервиса будет автоматически создана web-страница, показывающая, какие методы предоставляются web-сервиса, и позволяющая вызывать эти методы через web-браузер. Чтобы протестировать web-сервис, используйте web-браузер для перехода по адресу web-сервиса и класса, указанных на шагах 1 и 2 (например, `http://MyWebServer/WebService1/Service1.asmx`).

Листинг 3.32. Простой web-сервис

```
//Этот код следует вставить в класс Service,
// содержащийся в файле "Service.asmx".
//
//"[WebMethod]" – это атрибут метаданных,
// который указывает механизму web-сервиса на то,
// что данный метод должен быть доступным через web
[WebMethod]
public int AddTwoNumbers(int x, int y)
{
     return x + y;
}
```

3.3.3. Вызов web-сервиса с мобильного устройства

Вызов web-сервиса с мобильного устройства работает почти точно так же, как и вызов, выполняемый с настольного компьютера или сервера.

Замечание: Для выполнения кода на физическом или эмулированном устройстве используется логическая машина, отличная от машины разработки; даже если эмулятор и выполняется на той же физической машине, что и ваш web-сервер, имена соответствующих машин будут разными. Вследствие этого *нельзя* использовать URL //localhost/WebService1, чтобы задать с устройства местоположение web-сервиса; необходимо использовать фактическое имя хост-машины (например, //myDevMachine1/WebService1) так, как вызывали бы web-сервис с другого персонального компьютера.

Для вызова web-сервиса с мобильного устройства, необходимо выполнить следующие действия:

1. Запустите Visual Studio и создайте проект C# «Smart Device Application».
2. Выберите в меню Project (Проект) пункт «Add Web Reference» (Добавить web-ссылку). В результате этого на экране отобразится диалоговое окно, где можно найти web-сервис, на которую требуется сослаться. Введите URL-адрес web-сервиса, который был создан перед этим (например, `http://MyWebServer/WebService1/Service1.asmx`).
3. Перейдя в диалоговое окно «Add Web Reference», введите в текстовом окне «Web Reference Name» (Имя web-ссылки) текст `MyTestWebService1.exe`, а затем щелкните на кнопке «Add Reference» (Добавить ссылку). В результате этого будет загружен WSDL-документ, описывающий web-сервис, а в вашем проекте создан локальный прокси-класс, благодаря которому можно будет легко вызывать данный web-сервис.
4. Поместите на форму разрабатываемого приложения кнопку и дважды щелкните на ней. В результате этого фокус ввода переместится в код обработчика событий кнопки.
5. Введите код, показанный на листинге 3.33.

Листинг 3.33.

```
//Создать экземпляр локального прокси-объекта
  MyTestWebService.Service1 myWebService;
  myWebService = new MyTestWebService.Service1();
//Указать локальному прокси-объекту на
// необходимость вызова web-сервиса
  int sum = myWebService.AddTwoNumbers (2, 3);
//Отобразить результат вызова web-сервиса!
  System.Windows.Forms.MessageBox.Show(sum.ToString());
```

6. Запустите приложение и вызовите web-сервис.

Данный пример соответствует тестированию синхронного вызова web-сервиса. Автоматически сгенерированный класс обладает встроенными возможностями, которые позволяют это сделать. Для вызова web-сервиса в асинхронном режиме следует вызвать метод `myWebService.BeginAddTwoNumbers (...параметры...)`. Для каждого метода web-сервиса в локальном прокси-классе предусмотрен метод `Begin*`, предназначенный для асинхронного вызова web-сервиса, и метод `End*`, предназначенный для получения результатов этого вызова.

При вызове функции `Add` созданного web-сервиса будет использоваться XML файл, показанный в листинге 3.34.

Листинг 3.34.

```
POST /WebService1/Service.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
<soap12:Body>
    <AddTwoNumber xmlns="http://tempuri.org/">
          <x>int</x>
          <y>int</y>
     </AddTwoNumber>
</soap12:Body>
</soap12:Envelope>
```

3.3.4. Трудности использования web-сервисов на мобильных устройствах

Несмотря на то, что использование web-сервисов на мобильных устройствах во многом напоминает использование web-сервисов на настольных компьютерах и серверах, между этими двумя случаями

имеются важные различия. Описанию трудностей, которые либо являются специфичными, либо проявляются в более заметной степени в случае мобильных устройств, посвящен следующий раздел.

Использование cookie-файлов

Cookie-файл – это порция локальных клиентских данных устройства, принадлежащая определенному web-сайту и управляемая им. Принадлежащие web-сайту данные, которые содержатся в cookie-файле, передаются на сервер вместе с остальной частью HTTP-запроса. Например, каждый из серверов `www.MyWebAddress.com` и `www.YourWebAddress.com` может поддерживать свой cookie-файл на машине-клиенте, осуществляющей доступ к этим web-сайтам. Если вычислительное устройство и приложение поддерживают cookie-файлы на стороне клиента, то всякий раз, когда по web-адресу или его подадресу (например, `www.MyWebAddress.com/somepath/somepath1`) высылается HTTP-запрос, вместе с ним на сервер передаются и данные, содержащиеся в соответствующем cookie-файле.

Cookie-файлы часто применяются для хранения информации о предпочтительных установках каждого из клиентов web-сайтов. Передаваемая вместе с каждым запросом информация из cookie-файла представляет интерес для сервера постольку, поскольку она избавляет сервер (или совместно действующую группу серверов) от необходимости поддерживать "состояние сеанса" на стороне сервера и облегчает масштабирование web-приложения, позволяя ему не сохранять информацию о состоянии в промежутках времени между запросами.

Не исключено, что web-сайт, предоставляющий информацию о погоде в вашем городе или ваши четыре излюбленные биржевые сводки, использует cookie-файлы на стороне клиента либо для хранения информации об этом в явном виде, либо для хранения уникальных идентификаторов, позволяющих серверу находить соответствующую информацию в базе данных сервера. Cookie-файлы могут быть использованы для хранения как кратковременных данных рабочего сеанса, например, "списка покупок" в web-магазине, так и долговременных данных, которые хранятся на протяжении нескольких различных сеансов, например, информации о наиболее часто посещаемых пользователем web-страницах, посвященных биржевым сводкам или прогнозу погоды. Вместе с тем, использованию клиентских cookie-файлов свойственны некоторые серьезные ограничения:

- ***Cookie-файлы являются специфическими для клиента и машины.*** Если web-приложение использует cookie-файлы на стороне клиента, то при доступе пользователя к данному web-приложению

с другой машины они должны создаваться заново. Это означает, что информация о предпочтениях пользователя, хранящаяся в cookie-файлах, не переходит вместе с пользователем на другую машину.

- ***Использовать cookie-файлы не всегда безопасно.*** Web-приложения не должны хранить в cookie-файлах ценную информацию, поскольку она будет пересылаться в обоих направлениях при каждом вызове, а ее копия будет сохраняться на клиентской машине, где она становится доступной для злонамеренных хакеров через точки уязвимости на стороне клиента. Критически важная информация должна надежно храниться на сервере и передаваться в другие места лишь по мере необходимости.
- ***Передаваемые cookie-файлы дополнительно занимают часть полосы пропускания.*** Поскольку cookie-файлы передаются с каждым web-запросом, они используют часть полосы пропускания канала связи. При передаче по сетям мобильной телефонной связи эта дополнительная нагрузка приводит к увеличению длительности и стоимости передачи. Чем больше размер cookie-файла, тем большая часть полосы пропускания тратится понапрасну.
- ***Cookie-файлы имеют ограниченные размеры.*** Существуют определенные ограничения в отношении объема данных, которые могут храниться в cookie-файлах.

Кроме вышеперечисленных ограничений общего характера, использование cookie-файлов при работе с web-сервисами характеризуется еще одним недостатком – сложностью. Сеанс связи с web-сервисом можно рассматривать как последовательность определенных запросов, передаваемых между клиентом и сервером. Часто эти запросы можно рассматривать как вызовы методов с передачей параметров и последующим получением возвращаемых результатов. Использование cookie-файлов при вызове web-сервис представляет собой второй скрытый канал связи между клиентом и сервером, и это может приводить к некоторой путанице. В листинге 3.35 продемонстрирован вызов web-сервиса без использования cookie-файлов, тогда как листинг 3.46 соответствует тому же примеру, в котором вместо передачи некоторых параметров используются cookie-файлы.

Листинг 3.35. Вызовы web-сервиса с передачей параметров только явным образом

```
//0. Установить связь
int sessionID = someWebService.LogOn
     (userCredentials);
//
//...Выполнение другого многострочного кода...
//
```

```
//1. Вызвать web-сервиса и создать новый заказ
int orderlD = someWebService.CreateNewOrder
     (sessionID, userlnfo, productlnfo);
//...Выполнение другого многострочного кода...
//
//2. Подтвердить заказ серверу
someWebService.ConfirmPayment
     (sessionID, orderlD, paymentlnfo);
//
//...Выполнение другого многострочного кода...
//
//3. Подтвердить адрес доставки
someWebService.ConfirmShipping
     (sessionID, orderlD, shippingAddress);
//
//...Выполнение другого многострочного кода...
//
//4. Завершить оформление заказа
someWebService.FinalizeOrder (sessionID, orderlD);
```

В данной программе на шаге 1 создается новый заказ и возвращается новый идентификатор заказа (`orderID`), который будет использоваться в последующих вызовах. Этот номер заказа передается в каждый последующий запрос, поэтому вам должно быть ясно, что каждый из вызовов web-сервиса может идентифицировать обрабатываемый заказ при помощи переданного ему параметра `orderID`.

Вместо использования явного параметра `orderID` эту информацию можно передавать web-сервису при помощи cookie-файла, хранящегося на стороне клиента. В этом случае клиентский код должен выглядеть примерно так, как показано в листинге 3.36.

Листинг 3.36. Вызов web-сервиса путем неявной передачи параметров посредством cookie-файлов

```
//0. Установить связь
//Хотя этого и не видно, с сервера будет
// передан клиентский cookie-файл!
int sessionID =
     someWebService.LogOn(userCredentials);
//1. Вызвать web-сервис и создать новый заказ
//Хотя этого и не видно, на сервер будет
//передан клиентский cookie-файл!
//Хотя этого и не видно, с сервера будет
// передан клиентский cookie-файл!
someWebService.CreateNewOrder(userInfo, product Info);
//...Выполнение другого многострочного кода...
//
//2. Подтвердить заказ серверу
//Хотя этого и не видно, на сервер передается
```

```
//клиентский cookie-файл, содержащий "orderID".
someWebService.ConfirmPayment (paymentlnfo);
//...Выполнение другого многострочного кода...
//
//3. Подтвердить адрес доставки
// На сервер передается
// клиентский cookie-файл, содержащий "orderID".
someWebService.ConfirmShipping(shippingAddress);
//...Выполнение другого многострочного кода...
//. . .
//4. Завершить оформление заказа
//На сервер передается клиентский
//cookie-файл, содержащий "orderID".
someWebService.FinalizeOrder();
```

Приведенный выше код довольно прост, однако, о чем говорится в комментариях, имеется и второй канал связи, скрытый от программиста. Скрытые параметры передаются в обоих направлениях между клиентом и сервером посредством cookie-файлов. Этот факт является убедительным аргументом в пользу того, чтобы не использовать cookie-файлы на стороне клиента при проектировании web-сервисов. Гораздо лучше передавать все параметры, требуемые для вызова web-сервиса, явным образом, чем использовать для хранения этой информации непрозрачный второй канал.

Многие платформы мобильных устройств либо вообще не поддерживают клиентские cookie-файлы, либо эта поддержка существенно отличается от той, которая предлагается программными каркасами на настольных компьютерах. В частности, в .NET Compact Framework, выполняющейся на устройствах Smartphone, Pocket PC и Windows CE, автоматическая передача cookie-файлов вместе с вызовом web-сервиса не поддерживается. Если требуется, чтобы некоторые cookie-файлы были переданы сервером на устройство и возвращены на сервер вместе с последующим запросом, то необходимо написать код для чтения содержимого cookie-файлов из заголовков одного из ответов `HTTPWebResponse` и записи содержимого cookie-файлов в заголовки последующего запроса `HTTPWebRequest`. Для этого в случае вызова web-сервиса необходимо просмотреть и изменить прокси-код web-сервиса на стороне клиента, автоматически сгенерированный средой разработки Visual Studio. Эта задача не является неразрешимой, но потребует от выполнения дополнительной работы. В этом и состоит важное отличие в поддержке web-сервисов программными средами на устройствах и настольных компьютерах.

Несмотря на тот факт, что использовать cookie-файлы на стороне клиента при создании web-сервисов рекомендуется, они могут исполь-

зоваться некоторыми службами, например для хранения информации о входе пользователя в систему. Если web-сервис работает нормально, если вызывается на настольном компьютере, но ее вызовы с мобильного устройства заканчиваются непонятными сбоями, то не исключено, что виновником этих сбоев являются cookie-файлы. Если есть такая возможность, уточните у разработчика web-сервиса, используются ли в ней cookie-файлы; это всегда проще, чем пытаться самостоятельно восстановить причину происходящего. Если получить эту информацию от разработчика web-сервиса не удается, можно попытаться исследовать ситуацию пробным путем изменения политики обработки cookie-файлов на настольном компьютере; соответствующие изменения можно задать в обозревателе Internet Explorer, выбрав в меню Tools (Сервис) пункт Options (Свойства обозревателя) и перейдя в открывшемся диалоговом окне на вкладку Privacy (Конфиденциальность). Кроме того, если есть желание заняться разработкой низкоуровневого кода клиентов web-сервисов, то нужно изучить набор клиентских cookie-файлов, возвращенный вместе с ответом `HTTPWebResponse` на web-запрос. Если в зависимостях клиентских cookie-файлов имеются ошибки, то можно действовать одним из следующих способов:

1) обеспечить поддержку web-сервисом модели доступа, не требующей использования cookie-файлов, что неплохо сделать в любом случае,
2) создать web-сервис в виде оболочки на стороне сервера, которая играет роль посредника между мобильными устройствами и проблематично web-сервисом, или
3) написать для устройства собственный код, который явным образом осуществляет сборку cookie-файлов, возвращенных вместе с ответами любого web-сервиса, и упаковывает их в последующие web-запросы.

Проблемы первого вызова web-сервиса

Во время первого обращения мобильного приложения к web-сервису часто выполняется значительный объем дополнительной работы, что проявляется в увеличении времени задержки. При первом вызове web-сервиса должна быть выполнена следующая работа:

1. ***Может потребоваться загрузка кода***. Если XML, web-сервис, сеть и другие классы на стороне клиента еще не были загружены в память, то не исключено, что их необходимо будет загрузить и компилировать, прежде чем они смогут быть использованы для вызова web-сервиса. Для этого потребуется определенное время, которое может исчисляться несколькими секундами.

2. ***Может потребоваться поиск адреса web-сервиса***. Например, если вызывается web-сервис по адресу `www.myWebService.com`, то для обнаружения местонахождения сервера этот адрес должен быть преобразован в IP-адрес (например, 200.134.81.26). Для нахождения этого адреса DNS-серверу направляется запрос на преобразование web-адреса в IP-адрес. Выполнение этой операции требует определенного времени; запрос необходимо упаковать и переслать на DNS-сервер, после чего ваше мобильное приложение должно дождаться ответа и лишь после этого сможет установить фактическую связь с сервером, который предоставляет вызываемый web-сервис. Большинству мобильных устройств приходится локально кэшировать этот адрес, чтобы последующие запросы, направляемые на web-сервер, не требовали повторного проведения поиска соответствующего имени сервером DNS. Выполнение процедуры разрешения имен требует заметного времени и может стать основной причиной задержки при первоначальном вызове web-сервиса. Как правило, поиск локального сетевого имени (например, `/XmyLocalServer`) происходит быстрее, чем поиск имени во всемирной сети (например, `www.myWebServer.com`).

При измерении времени отклика web-сервиса полезно определять две разновидности этой характеристики:

1) время отклика при выполнении первого вызова, и
2) среднее время отклика для последующих вызовов web-сервиса.

Поскольку направляемые web-сервисам запросы связаны с ресурсами, находящимися вне сферы непосредственного контроля вашего приложения, никогда не помешает осуществлять эти вызовы в асинхронном по отношению к потоку выполнения пользовательского интерфейса режиме. Несмотря на это, будут встречаться случаи, когда пользователь мобильного приложения, который запрашивает информацию или пытается выполнить транзакцию, должен будет дожидаться ее завершения, прежде чем сможет продолжить работу. В этих случаях целесообразно сделать все возможное для того, чтобы ускорить передачу данных на сервер. Если приложению известно, что пользователю потребуется вызвать web-сервис, то имеет смысл сделать фоновый «фиктивный вызов» того же web-сервиса еще до того, как потребуется выполнять реальный запрос. В результате «фиктивного вызова» произойдет предварительная загрузка всех необходимых классов в память и кэширование IP-адресов, которые понадобятся во время выполнения последующих вызовов.

Проблемы передача больших объемов данных

Хотя и можно передать web-сервису массив, состоящий из 2000 целых чисел, или загрузить с web-сервиса массив данных этого же типа, никто этого не делает. Проблемы передача больших объемов данных посредством запросов web-сервисов неэффективна. Web-сервисы оптимизированы для обеспечения максимальной гибкости и дружественности к протоколу HTTP. По этой причине для передачи информации web-сервисы используют большие объемы связанных с этой информацией текстовых данных.

Например, число 32 можно представить в виде одного байта данных 00100000. Для передачи целого числа 32 в XML-формате его необходимо представить, скажем, как `<int>32</int>`, для чего требуется уже 14 байт данных; то же самое относится и к другим типам данных. Те самые свойства, благодаря которым web-сервис и XML обретают гибкость, делают их неэффективными в смысле объема передаваемых данных.

Если web-сервис должен передавать приложению большие объемы данных, то лучше всего организовать это так, чтобы она возвращала указатель на файл с двоичными данными, а не поток данных в виде XML. В качестве показательного примера можно привести web-сервис, возвращающий фото изображения.

Хотя web-сервис и может возвратить изображение в виде массива байтов или целых чисел в кодировке XML, гораздо эффективнее возвратить строку с URL-адресом, указывающим на двоичный файл (например, //somewebserver/someshare/somedir/ somefile.jpg), который может быть загружен мобильным приложением. Именно так действуют web-браузеры; они загружают текст в удобочитаемой форме и компонуют информацию в виде HTML-документа, содержащего ссылки на двоичные файлы изображения, которые должны быть встроены в макет. Очень важно тщательно продумывать, в каком виде следует перемещать данные, и оптимизировать этот процесс, если объемы данных велики.

Каждый запрос, посланный на сервер, означает необходимость начала, проведения и завершения диалога с сервером; следовательно, этот процесс сопровождается дополнительными накладными расходами по организации связи. Пять отдельных вызовов web-сервиса, каждый из которых требует передачи одного параметра, гораздо менее эффективны, чем один запрос, содержащий пять параметров. Кроме того, учитывая прерывистый характер сеансов мобильной связи, использование пяти мелких запросов вместо одного более крупного повышает вероятность сбоя в процессе диалога между устройством и сервером. Это означает необходимость написания сложной логики, позволяющей

выполнять необходимые восстановительные операции, если посередине такого диалога что-то пойдет не так. Применение одиночных вызовов web-сервисов позволяет уменьшить вероятность сбоев и упростить логику восстановления связи в подобных случаях.

В листинге 3.37 приведен пример неэффективной организации работы с web-сервисом с использованием многократных запросов и ответов.

Листинг 3.37. Неэффективная организация диалога с web-сервисом, в которой используется множество вызовов

```
//Создать и обработать заказ
//0. Установить связь
int sessionID = someWebService.LogOn(userCredentials);
//1. Вызвать web-сервис и создать новый заказ
int orderID = someWebService.CreateNewOrder(sessionID,
     userlnfo, productlnfo);
//2. Вызвать web-сервис и передать информацию о платеже
someWebService.ConfirmPayment(sessionID, orderID,
     paymentlnfo);
//3. Вызвать web-сервис и передать информацию о доставке
someWebService.ConfirmShipping(sessionID, orderID,
     shippingAddress);
//4. Вызвать web-сервис и завершить оформление заказа some-
WebService.FinalizeOrder(sessionID, orderID);
```

В листинге 3.38 представлен пример пакетной организации того же диалога, когда вся необходимая обработка осуществляется при помощи одного цикла "запрос/ответ". При любом удобном случае старайтесь уменьшать количество запросов, объединяя несколько мелких запросов в один более емкий.

Листинг 3.38. Группирование запросов в одном вызове web-сервиса

```
//-----------------------------------------------
//Создать и обработать заказ посредством группового запроса,
//включающего:
// 0. Начало связи
// 1. Создание нового заказа
// 2. Подтверждение платежа
// 3. Подтверждение доставки
// 4. Завершение оформления заказа
// -----------------------------------------------
//Сделать все за один раз
someWebService.BatchCreateOrder (userCredentials, userlnfo,
     paymentlnfo, shippingAddress);
```

В листинге 3.39 показан пример кода, который загружает с сервера двоичный файл и сохраняет его локально на устройстве. Этот код может пригодиться вам при загрузке с сервера файлов, подобных файлам изображений.

Листинг 3.39. Код для загрузки файла с web-сервиса

```
//Выполняем синхронную загрузку файла с web-сервера
//и сохраняем его в локальной файл
// [in] httpWhereFrom: URL-адрес файла
//      (например, "http://someserver/somefile.jpg")
// [in] filenameWhereTo: Место, куда необходимо записать
// файл (например, "\\localfile.jpg")
public void downloadFileToLocalStore(string httpWhereFrom,
     string filenameWhereTo)
{
  System.IO.FileStream myFileStream = null;
  System.IO.Stream myHTTPResponseStream = null;
  System.Net.WebRequest myWebRequest = null;
  System.Net.WebResponse myWebResponse = null;
//Если файл, который нужно записать, уже существует, удалить
  if(System.IO.File.Exists(filenameWhereTo) == true)
     System.IO.File.Delete(filenameWhereTo);
  try
  { // Создаем web-запрос
    myWebRequest =
        System.Net.HttpWebRequest.Create(httpWhereFrom);
    // Получаем ответ
    myWebResponse = myWebRequest.GetResponse();
    // Получаем поток для ответа
    myHTTPResponseStream=myWebResponse.GetResponseStream();
    // Создаем локальный файл, в который будет
    // направлен поток ответа
    myFileStream =
        Systern.IO.File.OpenWrite(filenameWhereTo);
    //Этот размер буфера является настраиваемым
    const int buffer length = 4000;
    byte [] byteBuffer = new byte[buffer length];
    int bytesIn;
// Читаем файл и направляем поток данных в локальный файл
    do {
     //Читаем передаваемые данные
     bytesln = myHTTPResponseStream.Read(byteBuffer, 0,
        buffer_length);
     //Записываем данные в файл
     if(bytesIn != 0)
         myFileStream.Write(byteBuffer, 0, bytes In );
    } while (bytesIn != 0);
  }
  catch (Exception myException) //Сбой при загрузке!
  {
  //Что-то случилось. Освободить ресурс!
    attemptCleanup_ThrowNoExceptions (myFileStream,
```

```
        myHTTPResponseStream, myWebResponse) ;
// После освобождения, повторно генерируем
// исключение, чтобы сообщить о сбое!
    throw myException;
  }
//Загрузка прошла успешно! Закрыть все ресурсы!
  try
  { //Стандартная процедура закрытия ресурсов
    myFileStream.Close(); myFileStream = null;
    myHTTPResponseStream.Close();
    myHTTPResponseStream = null;
    myWebResponse.Close(); myWebResponse = null;
  }
  catch (Exception myException)
  { //Был сбой в процессе закрытия ресурса!
    //Что-то случилось. Освободить ресурс!
    attemptCleanup ThrowNoExceptions {myFileStream,
        myHTTPResponseStream, myWebResponse) ;
// После освобождения, повторно генерируем
// исключение, чтобы сообщить о сбое!
    throw myException;
  }
//Успешное выполнение!
}

//-----------------------------------------------
//Пытаемся закрыть и освободить все объекты
//Перехватывает любое вырабатываемое исключение.
//-----------------------------------------------
void attemptCleanup ThrowNoExceptions (
        System.IO.FileStream myFileStream,
        System.IO.Stream myHTTPResponseStream,
        System.Net.WebResponse myWebResponse)
{
  if (myFileStream != null) {
  try{ myFileStream.Close (); )
     catch{} //Не выполнять никаких действий.
  }
  if(myHTTPResponseStream != null){
    try{myHTTPResponseStream.Close();}
    catch{} //Не выполнять никаких действий.
  }
  if(myWebResponse != null) {
    try {myWebResponse.Close(); }
    catch{} //Не выполнять никаких действий.
  }
} //конец функции
```

Web-службы доказали свою полезность, и их использование для организации взаимодействия между различными системами будет только расширяться, но в интересах сохранения высокой производительности мобильных приложений действовать при этом надо осмотрительно. Используемые мобильными приложениями коммуникационные средства предоставляют богатое разнообразие возможностей выбора для разработчиков программного обеспечения. Однако это же обстоятельство повышает и ответственность разработчика за правильность сделанного им выбора с точки зрения соответствия этого выбора запросам пользователей мобильного приложения и требованиям экономичности.

3.4. Разработка мобильных web-приложений

С помощью мобильных элементов управления технологии ASP.NET можно создавать web-приложения для мобильных устройств, используя ту же модель приложения, что и в разработках для настольных компьютеров. Мобильные элементы управления предназначены для широкого спектра устройств – сотовых телефонов, пейджеров, КПК и других персональных помощников, таких как Palm и BlackBerry. Для поддержки мобильных технологий модель Web Forms в ASP.NET была расширена в двух направлениях, а именно поддержка устройств и поддержка специализированных элементов управления. Все поддерживаемые устройства перечисляются в новом разделе файла `machine.config`, куда могут добавлять сведения о новых устройствах и элементах управления.

Мобильные приложения создаются с использованием мобильных элементов управления ASP.NET и ее объектной модели. Основываясь на данных конфигурации, подсистема Mobile Web Forms определяет, от какого устройства поступил текущий запрос, и осуществляет рендеринг с применением соответствующего языка разметки и необходимой логики. Мобильные элементы управления абстрагируют подмножество серверных элементов управления ASP.NET и интеллектуально производят рендеринг приложения для целевого устройства. Кроме того, существует несколько специализированных мобильных элементов, к числу которых относится, например, `PhoneCall`.

В данном разделе пособия приведен обзор мобильных элементов управления ASP.NET, рассмотрены основы технологии разработки мобильных приложений и коротко описаны функции Visual Studio для разработки таких приложений.

3.4.1. Обзор мобильных элементов управления

Мобильные элементы управления ASP.NET – это серверные элементы управления, а следовательно, их атрибуту `runat` должно быть при-

своено значение `server` (`runat = «server»`). Они выполняются на сервере и поэтому могут использовать возможности .NET Framework несмотря на небольшой объем памяти мобильных устройств. Все мобильные элементы управления наследуют класс `MobileControl`, который, в свою очередь, является производным от класса `Control`. Серверные и мобильные элементы управления имеют общий набор свойств, методов и событий. Мобильные Web-формы наследуют класс `MobilePage`.

Архитектура мобильных элементов управления

Мобильные элементы управления обеспечивают возможность адаптации стандартных элементов управления ASP.NET к устройству, от которого получен запрос. Ключевым моментом их работы является идентификация браузера, которая играет здесь гораздо более важную роль, чем при использовании обычных элементов управления.

Мобильные и классические серверные элементы управления

Рендеринг классических серверных элементов управления выполняется по-разному в зависимости от типа целевого клиента (настольного браузера). Однако, хотя результирующая разметка для разных целевых клиентов может сильно различаться, эти различия касаются главным образом стиля.

Классические серверные элементы управления всегда генерируют HTML или XHTML-код. Хотя между стандартами HTML 3.2 и HTML 4.0 и между этими стандартами и их реализацией в конкретных браузерах имеются некоторые различия, все это не идет ни в какое сравнение с разнообразием синтаксисов и диалектов разметки, использующихся на мобильных устройствах. Поэтому функция определения браузеров мобильных устройств реализована на уровне системы, тогда как в приложениях для настольных браузеров ее реализация оставлена на усмотрение разработчика. Мобильная исполняющая среда ASP.NET анализирует строку пользовательского агента и выбирает соответствующий адаптер мобильного устройства. Вот почему мобильные элементы управления приспосабливаются к возможностям конкретного устройства, направившего запрос.

Набор поддерживаемых беспроводных устройств довольно разнообразен – от персональных цифровых помощников с полнофункциональными браузерами до сотовых телефонов с миниатюрными экранами. Поэтому не только разметка, но и структура приложения должны существенно изменяться в зависимости от типа устройства. В общем случае средство разработки мобильных приложений должно абстрагировать многие аспекты клиентской среды, такие как различие

языков разметки, зависящее от реализации одного и того же языка, разные форм-факторы, в частности разные размеры дисплея и программируемых кнопок. Мобильные элементы ASP.NET берут на себя задачу абстрагирования, позволяя вам сконцентрироваться на логике приложения.

Разные языки разметки

Мобильные элементы управления поддерживают несколько языков разметки. Наиболее популярные из них перечислены в табл. 3.1.

Таблица 3.1

Языки разметки, поддерживаемые мобильными элементами управления

Язык разметки	Описание
Compact HTML 1.0	Подмножество HTML 3.2, специально разработанное для мобильных телефонов, а также других устройств с питанием от батареи и малым объемом памяти. Сокращенным названием данного языка является cHTML. Язык широко применяется в японских устройствах I-Mode.
HTML 3.2	Относительная старая версия HTML, поддерживаемая такими карманными компьютерами, как Pocket PC 2002, Palm и BlackBerry
WML 1.x	Wireless Markup Language (язык разметки беспроводных устройств) является одним из самых популярных языков беспроводных устройств. Он разработан в рамках инициативы более широкого масштаба, целью которой было определение прикладного протокола беспроводных устройств – широко известного Wireless Application Protocol (WAP). Язык WML позволяет программировать клавиши мобильного телефона, в частности цифровую и программную клавиатуры.
XHTML	Это XML-версия HTML, являющаяся ключевым элементом стандарта WAP 2.0

Иерархия мобильных элементов управления

Мобильные элементы управления ASP.NET логически подразделяются на пять категорий: 1) контейнеры, 2) текст, 3) условия проверки, 4) списки и 5) другие разные элементы. Все они в алфавитном порядке перечислены в табл. 3.2. Некоторые из них очень похожи на элементы управления Web Forms. Однако все мобильные элементы управления отличаются адаптивным рендерингом. Тема адаптивного рендеринга будет подробно рассмотрена в отдельном разделе.

Таблица 3.2

Мобильные элементы управления ASP.NET

Элемент управления	Описание
AdRotator	Обеспечивает циклическую смену рекламных баннеров подобно тому, как это делает одноименный элемент управления ASP.NET.
Calendar	Дает пользователю возможность выбрать дату, в точности как одноименный элемент управления ASP.NET
Command	Подобно элементу управления *Button* из ASP.NET по щелчку генерирует событие возврата формы
Compare Validator	Идентичен одноименному элементу управления ASP.NET
Custom Validator	Идентичен одноименному элементу управления ASP.NET
DeviceSpecific	Обеспечивает возможность адаптировать внешний вид элементов управления *Form* и *Panel* к определенному типу аппаратных устройств.
Form	Подобен элементу управления *HtmlForm* из ASP.NET, однако на мобильной странице может выводиться несколько форм
Image	Идентичен одноименному элементу управления ASP.NET
Label	Идентичен одноименному элементу управления ASP.NET
Link	Представляет гиперссылку на другую форму мобильной страницы или на произвольный URL. Подобен элементу управления *HyperLink* из ASP.NET.
List	Подобен элементам управления ASP.NET *Repeater* и *DataList,* применяет шаблоны к связанным данным. Поддерживает специфические шаблоны различных устройств
ObjectList	Подобен элементу управления ASP.NET *DataGrid,* выводит строки таблицы данных и поддерживает несколько команд
Panel	Подобно элементу управления ASP.NET *Panel* служит для группировки и организации элементов управления

Окончание табл. 3.2

Элемент управления	Описание
PhoneCall	Для устройств, поддерживающих телефонию, представляет ссылку на номер телефона, по которому можно позвонить. Подобен тэгу *mailto* гиперссылки HTML
RangeValidator	Идентичен одноименному элементу управления ASP.
RegularExpression Validator	Идентичен одноименному элементу управления ASP.NET
RequiredFieldValidator	NET Идентичен одноименному элементу управления ASP.NET
SelectionList	Выводит список элементов, связанных с данными. В зависимости от того, он будет сконфигурирован, может вести себя как раскрывающийся список или список флажков. Выбор элемента в таком списке не вызывает автоматического возврата формы, но при следующем возврате формы генерируется серверное событие
StyleSheet	Невидимый элемент управления для централизованного определения стилей, применяемых к другим элементам управления. Может содержать несколько определений стилей
TextBox	Представляет однострочное текстовое поле; режим «только чтение» не поддерживает
TextView	Используется для вывода больших блоков текста, поддерживает простейшее форматирование текста и листание
ValidationSummary	Подобен элементу управления *ValidationSummary* из ASP.NET, выводит результаты проверки в отдельной форме

Почти все мобильные элементы управления, кроме `PhoneCall` и *TextView,* имеют аналоги среди серверных элементов управления ASP.NET. Интерфейс программирования проверочных элементов практически такой же, как в ASP.NET, с той лишь разницей, что не все устройства поддерживают клиентскую проверку.

На рис. 3.31 показана схема взаимосвязей элементов управления. Элементы, представленные белыми прямоугольниками – это абстрактные классы.

Прежде чем переходить к подробному изучению характеристик каждой группы элементов управления, вам нужно кое-что узнать о структуре мобильной страницы ASP.NET.

3.4.2. Мобильные страницы ASP.NET

Базовым классом всех мобильных Web-страниц ASP.NET является `MobilePage`, наследующий класс ASP.NET `Page`. Класс разрабатываемой мобильной страницы наследуется от `MobilePage` или от класса, производного от него. Все мобильные элементы управления интерпретируются как специализированные (custom controls) и должны явно регистрироваться на странице. Рассмотрим исходный код демонстрационной мобильной страницы.

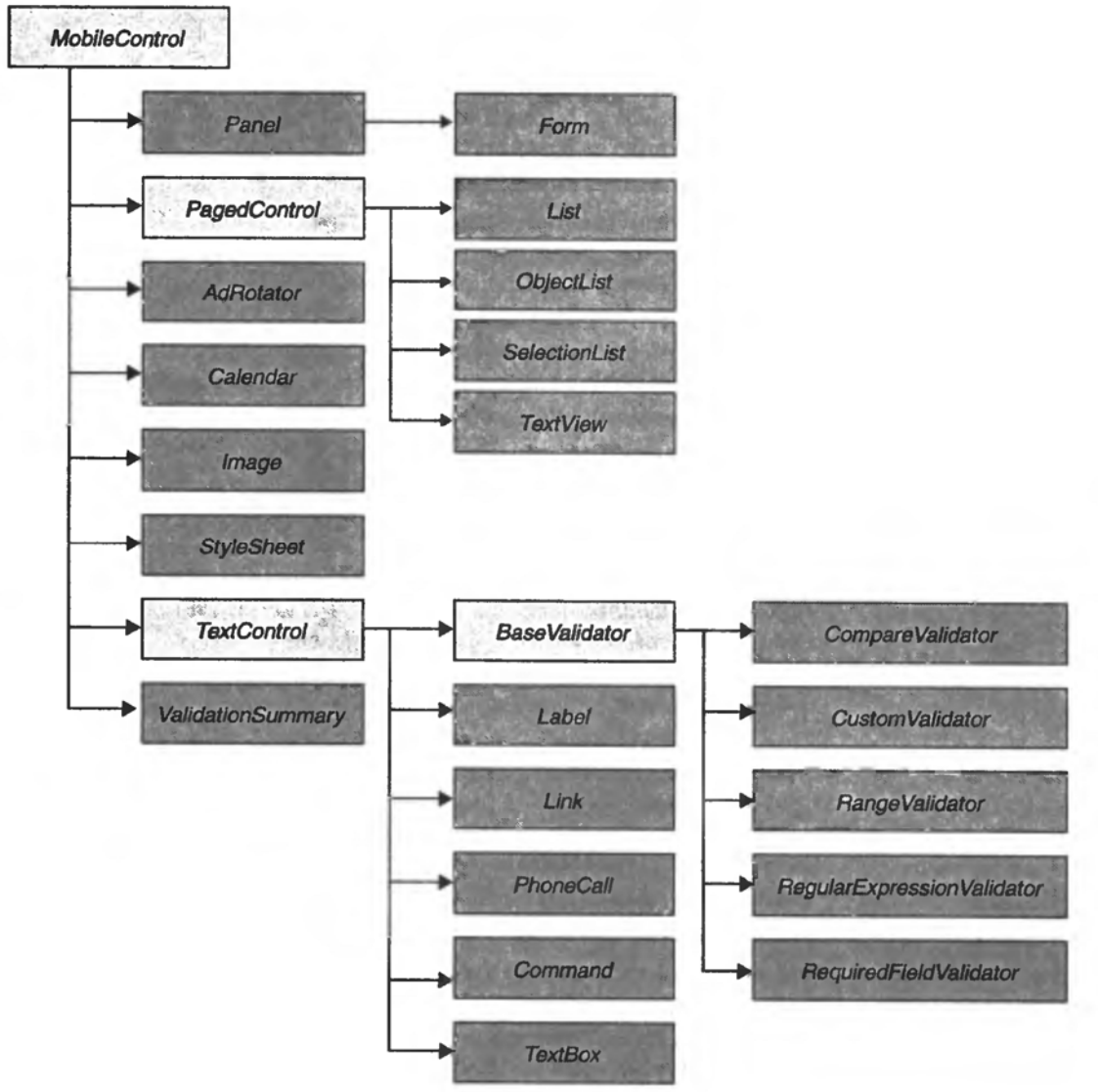

Рис. 3.31. Иерархия классов мобильных элементов управления ASP.NET

Пример мобильной страницы

Данная web-страница выводит название языка разметки, поддерживаемого беспроводным устройством (листинг 3.40).

Листинг 3.40.

```
<%@ Page Language="C#" CodeFile="Hello.aspx.cs" Inher-
its="Hello" %>
<%@ Register TagPrefix="mobile"
     Namespace= "System.Web. I.MobileControls"
     Assembly="System.Web.Mobile" %>
<%@ Import Namespace="System.Web.Mobile" %>
<html xmlns="http://www.w3.org/1999/xhtml" >
<body>
     <mobile:Form ID="Form1" runat="server"> Language is
     <mobile:Label runat="server" font-bold="true"
          id="theBrowser" />
</mobile:Form>
</body>
</html>
```

Страница содержит элемент управления `Label`, выводящий полужирным шрифтом заданный текст. Обработчик события загрузки страницы `Load` определяет предпочтительный для данного устройства тип рендеринга, считывает из конфигурационного файла соответствующие установки и выводит их. Класс `MobileCapabilities` определен в пространстве имен `System.Web. Mobile`, а все элементы управления – в пространстве имен `System.Web.UI.MobileControls`. Код обработки события показан в листинге 3.41.

Листинг 3.41.

```
public partial class Hello :
     Sys-
tem.Web.UI.MobileControls.MobilePage
{
   protected void Page Load(object sender,
   EventArgs e){
      MobileCapabilities mobCaps =
           (MobileCapabilities)Request.Brow
     ser;
      theBrowser.Text        =        mob-
      Caps.PreferredRenderingType;
   }
}
```

При просмотре на эмуляторе мобильного устройства, использующего браузер Openwave 7.0 (информацию о нем можно найдете по адресу `http://odn.openwave.com`), страница выглядит так, как показано на рис. 3.32.

Рис. 3.32. Мобильное приложение в эмуляторе Openwave

Мобильная страница определяет возможности устройства и генерирует разметку на языке WML 1.1, показанную в листинге 3.42.

Листинг 3.42.

```
<wml>
  <head>
    <meta http-equiv="Cache-Control"
       content="max-age=0" />
  </head>
  <card>
    <do type="accept"><noop /></do>
    <p>Language is <b>wml11</b></p>
  </card>
<wml>
```

Интересно, что ту же страницу можно открыть и в Internet Explorer. В таком случае целевым языком будет HTML 3.2. В данном случае вывод, направляемый целевому устройству, совсем иной (листинг 3.43).

Листинг 3.43.

```
<html> <body>
<form id="Form1" name="Form1" method="post" ac-
tion="Hello.aspx?__ufps=149742">
     <input type="hidden" name="__VIEWSTATE" value="/wEXA
     ... CVL">
     <input type="hidden" name="__EVENTTARGET" value="">
     <input type="hidden" name="__EVENTARGUMENT" value="">
     <script language=javascript><!--
          function _doPostBack(target, argument){
               var theform = document.Form1
               theform.__EVENTTARGET.value = target
               theform.__EVENTARGUMENT.value = argument
               theform.submit()
          }
     // -->
     </script>
     Language is <b>html32</b>
</form>
</body></html>
```

Объект `MobilePage` может содержать элементы управления только двух видов: `Form` и `StyleSheet`. Текст, размещенный вне формы, игнорируется. Если вне формы размещены другие элементы управления, компилятор выдает сообщение об ошибке. Весь необходимый текст и элементы управления должны размещаться внутри формы. Хотя мобильная страница может содержать и более одной формы, только одна из них видима в каждый конкретный момент времени. Применять таблицу стилей не обязательно, но если есть желание ее использовать, то она должна быть у страницы одна.

Разбивка на страницы

Мобильные элементы управления ASP.NET позволяют автоматически разбивать содержимое формы на небольшие части. Этот механизм называется *разбивкой на страницы* (pagination). При его использовании сегменты контента автоматически форматируются по размеру экрана целевого устройства. В конец результирующей клиентской страницы помещаются типичные элементы пользовательского интерфейса, предназначенные для перехода от одной части страницы к другой.

По умолчанию разбивка форм на страницы отключена. Чтобы включить ее, нужно задать свойству `Paginate` объекта `Form` значение *true.* Следует отметить, что, хотя данную функцию можно активизировать и для отдельных элементов управления, ваша установка не будет иметь никакого эффекта, если свойство `Paginate` формы-контейнера содержит *false.* У элемента управления `Form` есть и другие свойства, управляющие разбивкой на страницы, к их числу относятся `PageCount`, `CurrentPage` и `PagerStyle`. Существует возможность включить разбивку на страницы для отдельного элемента формы через свойство этой формы `ControlToPaginate`, которому нужно присвоить идентификатор элемента управления.

Главная цель разбивки на страницы – избежать перегрузки памяти устройства слишком большими страницами. Особенно эффективен данный метод для элементов управления, которые выводят на экран большой объем данных, а вот для форм, содержащих интерактивные элементы управления и поля ввода, его применение не является необходимым. Разбивка на страницы реализуется путем разделения формы или элемента управления на части.

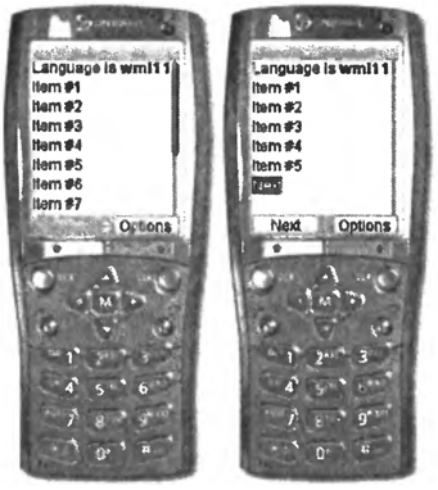

Рис. 3.33. Результат разбивки формы мобильного приложения на страницы

Некоторые элементы управления, например `List`, управляют разбивкой на страницы сами и сами решают, как лучше разделить свой вывод на части. Форма может содержать элементы-контейнеры, в которых размещаются другие элементы управления; чтобы такой контейнер мог быть разделен на части, его дочерние элементы управления должны помещаться на странице. На рис. 3.33 показан вывод формы с отключенной разбивкой

на страницы (слева) и рядом вывод той же формы, когда данная функция включена.

Замечание: Открыв на мобильном устройстве демонстрационную страницу, показанную на рис. 3.49, не удивляйтесь, если, щелкнув ссылку Next, получите сообщение об ошибке. В таком случае вам нужно отключить работу с cookie.

Жизненный цикл мобильной страницы

Жизненный цикл мобильной страницы ASP.NET, этапы которого перечислены ниже, практически идентичен жизненному циклу обычной страницы Web Forms. Он включает те же события, хотя поведение системы до и после их генерирования разное.

- **Инициализация страницы.** На этом этапе определяется адаптер устройства, который будет использоваться при формировании вывода страницы, и устанавливается свойство `Adapter` класса `MobilePage`. Поиск адаптера начинается с файла `machine.config`, затем производится в файле `web.config` из корневого каталога сайта, а уж потом – в файлах `web.config`, расположенных ниже по иерархии. Адаптер выбирается на основе характеристик текущего устройства. Обратите внимание: с целью повышения производительности адаптер кэшируется, поэтому его поиск для каждого пользовательского агента производится один раз.
- **Загрузка состояния представления.** На данном этапе восстанавливается сохраненное состояние страницы. Никакие события с этим этапом не связаны.
- **Загрузка данных возврата страницы.** Страница загружает входящие данные формы, кэшированные в объекте `Request`, и соответствующим образом обновляет свои свойства. Никакие пользовательские события с данным этапом не связаны.
- **Загрузка пользовательского кода.** Страница готова к выполнению инициализационного кода, связанного с ее логикой и поведением. Она генерирует событие `Load` и загружает информацию с учетом специфики адаптера устройства. Если есть желание управлять этим этапом, то можно реализовать обработчик данного события, как для страницы, так и для адаптера.
- **Отправка уведомлений об изменении возвращенных данных формы.** Элементы управления, входящие в состав страницы, генерируют события изменения, если их состояние изменилось со времени последнего возврата формы. Для того чтобы элемент управления мог генерировать на этом этапе событие изменения состояния, в нем нужно реализовать интерфейс `IPostBackData-`

`Handler`. Пользовательские события на данном этапе не генерируются.

- **Обработка события возврата формы.** Страница выполняет код, связанный с событием, которое вызвало возврат формы.
- **Предрендеринг.** Перед рендерингом вывода код получает последнюю возможность внести какие-либо изменения. Событие, с которым разработчик может связать соответствующий код мобильной страницы или адаптера устройства, называется `PreRender`. Именно на этом этапе производится разбивка на страницы. Вывод страницы формируется с учетом параметров разбивки.
- **Сохранение состояния представления.** Состояние страницы сериализуется в строку, которая затем сохраняется – обычно в виде скрытого поля. Пользовательские события на данном этапе не генерируются.
- **Рендеринг страницы.** Страница генерирует вывод, который будет направлен клиенту. За вывод дочерних элементов управления в нужном порядке отвечает адаптер.
- **Выгрузка страницы.** На этом этапе адаптер устройства выполняет необходимые финальные операции. Событие `Unload` доступно и странице, и адаптеру устройства.

Если сравнить процессы обработки запроса мобильной и настольной страниц в ASP.NET 2.0, то обнаружится различие, связанное главным образом с этапом инициализации – для мобильных страниц выбирается подходящий адаптер устройства, который затем используется для их рендеринга. В ASP.NET 2.0 это различие не столь существенно, поскольку для настольных страниц также применяется адаптивный рендеринг.

Адаптеры устройств

Благодаря применению адаптеров устройств программист может разрабатывать мобильное приложение, не ориентируясь на конкретное устройство. При обработке конкретного запроса адаптер транслирует высокоуровневое описание пользовательского интерфейса в слой представления, подходящий для того устройства, от которого этот запрос поступил. Можно провести интересную аналогию между драйверами устройств из Microsoft Windows и адаптерами мобильных устройств из ASP.NET. И те и другие генерируют команды, понятные конкретному устройству, только адаптеры выводят их в форме разметки. Адаптеры – это своего рода мосты между отдельными мобильными элементами управления и целевыми устройствами. Для каждого конкретного устройства мобильный элемент управления может иметь уникальный адаптер, и каждый экземпляр элемента управления связывается со сво-

им экземпляром адаптера. Наличие адаптера является ключевым условием рендеринга мобильных элементов управления. В общем случае для каждого поддерживаемого устройства могут быть определены следующие классы:

- **Адаптер страницы.** Будучи связанным со страницей, данный класс реализует функции, связанные с состоянием представления и возвратом формы, и включает методы, выполняющие сохранение и загрузку данных состояния элементов управления, специфических для конкретного устройства, в частности связанных с разбивкой на страницы и активной формой. Также данный адаптер отвечает за подготовку ответа клиенту и рендеринг каркаса страницы. Адаптер страницы реализует интерфейс `IPageAdapter`, относящийся к API поддержки мобильных устройств.
- **Адаптер формы**. Будучи связанным с элементом управления `form`, данный адаптер предоставляет методы, управляющие интерактивностью формы. Кроме того, он должен включать методы для адаптации универсальной формы к конкретному устройству. Например, он может разбивать поля формы на страницы или объединять их в меню. Наконец, адаптер формы должен быть способен вывести ее каркас.
- **Адаптер элемента управления**. Это базовый класс адаптеров элементов управления, также являющийся базовым и для адаптеров страницы и формы. Он предоставляет методы для рендеринга элементов управления, генерирующих события возврата формы, и для рендеринга свойств, определяющих стили.
- **Класс для записи текста (text writer)**. Данный класс не является адаптерным, но работает в тесном сотрудничестве с адаптерами. Он наследует класс `HtmlTextWriter`. Поскольку он выполняет для элемента управления весь необходимый рендеринг, экземпляр данного класса создается для каждого адаптера. Обычно класс для записи текста содержит вспомогательные методы, в частности, предназначенные для кодирования данных.

В состав ASP.NET 2.0 входит несколько адаптеров устройств – по одному для каждого языка разметки, cHTML, XHTML, HTML и WML.

3.4.3. Контейнерные элементы управления

Среди мобильных элементов управления ASP.NET есть два контейнера, `Panel` и `Form`, которые могут содержать другие элементы управления. Элемент `Form` может выполнять возврат данных серверу; `Panel`, являющийся просто средством группировки связанных между собой элементов управления, такой функцией не обладает. Панели мо-

гут быть вложенными, и в форме может содержаться любое их количество; формы же включать одна в другую нельзя.

Элемент управления Panel

Приведенный в листинге 3.44 код выводит две панели, содержащие текст с минимальным форматированием. Следует отметить, что не каждый атрибут поддерживается всеми существующими устройствами. Например, сотовые телефоны игнорируют цвета фона и текста.

Листинг 3.44.

```
<mobile:Form runat="server">
  <mobile:Panel runat="server" Font-Bold="true"
          Alignment="Right">
     <mobile:Label runat="server">
Пособие по программированию.
     </mobile:Label>
   </mobile:Panel>
   <mobile:Panel runat="server" BackColor="yellow">
     <mobile:Label runat="server"> Иванов А.И.
     </mobile:Label>
   </mobile:Panel>
</mobile:Form>
```

Панель может содержать любые мобильные элементы управления ASP.NET, кроме `MobilePage`, `Form` и `StyleSheet`. Сама она не имеет визуального представления, и все стили, которые для нее будут заданы, применяются к размещенным на ней элементам. Размещение дочерних элементов и их стили могут быть изменены адаптером устройства.

Заметьте, что элемент управления `Panel` может также использоваться и для выделения места под динамически генерируемые дочерние элементы управления.

Элемент управления Form

Элемент управления `Form` является самым внешним контейнером элементов управления, размещенных на мобильной странице. Он предоставляет такие часто используемые свойства, как `Action` и `Method`. Свойство `Action` по умолчанию содержит пустую строку, в результате чего при возврате формы задается исходный URL адрес. При активизации и деактивизации форма генерирует события. Страница может содержать несколько форм, но в каждый конкретный момент в браузере выводится только одна из них (листинг 3.45).

Листинг 3.45.

```
<mobile:Form id="Form2" runat="server">
     <mobile:Label runat="server"
```

```
          text="Вторая страница информации" />
     <mobile:Link runat="server" NavigateURL="#Form1"
          text="Back" />
</mobile:Form>
```

Установка текущей формы выполняется посредством свойства `ActiveForm`, доступного и для чтения. Когда на одной странице активны несколько форм, для перехода между ними используется элемент управления `Link`.

Для того чтобы элемент управления указывал на нужную форму, его свойству `NavigateURL` следует присвоить ее идентификатор с префиксом в виде символа #.

3.4.4. Списочные элементы управления

Мобильными эквивалентами традиционных списочных и итеративных элементов управления ASP.NET являются элементы управления `List`, `ObjectList` и `SelectionList`. Элемент `List` выглядит как `Repeater` и `DataList`. Он выводит статический или связанный с данными список элементов. Элемент `SelectionList` необходим для того, чтобы пользователь мог выбирать в списке более одного элемента управления. Что касается `ObjectList`, то он служит для отображения информации из базы данных в табличном формате. Отметьте, что он всегда связан с данными, тогда как два других элемента управления могут выводить списки статичных или программно генерируемых элементов.

Элемент управления `List`

Элемент управления `List` направляет на устройство последовательность элементов. Он может работать в статическом или интерактивном режиме. В статическом режиме данный элемент генерирует статический список чисто текстовых элементов. В интерактивном режиме получаются элементы, по щелчку на которых генерируются события. Если написан обработчик события `ItemCommand`, элемент интерактивен, в противном случае он статичен. Когда свойство `ItemsAsLinks` установлено равным `true`, элементы списка превращаются в гиперссылки. В таком случае событие `ItemCommand` не генерируется, а в ответ на щелчок пользователя просто выполняется переход по заданному URL. Этот URL присваивается свойству `DataValueField`, а текст гиперссылки задается в свойстве `DataTextField`.

Код на листинге 3.46 принадлежит странице, которая выводит список городов и позволяет пользователю выбрать один из них. Она состоит из двух форм: одна из них содержит список, а другая – информацию, которая выводятся после выбора элемента в этом списке.

Листинг 3.46.

```
<mobile:Form id="IntroForm" runat="server">
   <b>Куда вы хотите поехать сегодня?</b>
   <mobile:List runat="server" id="Cities"
         OnItemCommand="List_Click" >
      <item Text="Rome" Value="€10" />
      <item Text="New York" Value="$500" />
      <item Text="London" Value="€200" />
      <item Text="Paris" Value="€350" />
      <item Text="Sydney" Value="$1200" />
   </mobile:List>
</mobile:Form>
<mobile:Form runat="server" id="ResultsForm">
     <mobile:Label runat="server" id="Info"/>
</mobile:Form>
```

Класс отделенного кода выводит код для переключения в результирующую форму (листинг 3.47).

Листинг 3.47.

```
protected void List_Click(object sender,
     ListCommandEventArgs e)
{
     string msg = String.Format("Going to {0} for {1}.",
          e.Listltem.Text, e.Listltem.Value);
     Info.Text = msg;
     ActiveForm = ResultsForm;
}
```

На рис. 3.34 это приложение показано в действии. Список городов генерируется элементом управления `List`, а наличие обработчика `ItemCommand` обеспечивает возможность выбора элемента списка.

Рис. 3.34. Демонстрационное приложение

Замечание: *Будьте внимательны, когда используете символ $ в мобильных приложениях. В WML этот символ имеет особое значение – он предназначен для идентификации переменных. Однако мобильные элементы управления ASP.NET автоматически обрабатывают его вхождения в исходных файлах и правильно осуществляют его рендеринг, дублируя этот символ: $$.*

Элемент управления `List` можно связать с источником данных только посредством свойства `DataSource` и объектов, реализующих интерфейс `IEпитerable`. Мобильные элементы

управления не поддерживают элементов управления, представляющих источники данных, и не имеют свойства DataSourceID.

Элемент управления List можно связать всего с двумя столбцами источника данных, один будет привязан к свойству Text элемента списка, а другой – к свойству Value. Соответствующие установки задаются в свойствах DataTextField и DataValueField элемента управления List. Если требуется, чтобы текст элемента списка формировался на основе большего числа полей источника данных, определите обработчик события ItemDataBind (листинг 3.48).

Листинг 3.48.

```
void OnItemDataBind(object sender, ListDataBindEventArgs e)
{
     e.ListItem.Text = String.Format ("{0} - ${1}",
          DataBinder.Eval(e.Dataltem, "city"),
          DataBinder.Eval(e.Dataltem, "price"));
}
```

Для тех устройств, у которых возможности рендеринга сложнее, можно задавать шаблоны, по которым настраивается представление элементов данных. В таком режиме элемент управления List функционирует подобно серверному элементу управления ASP.NET Repeater.

Элемент управления ObjectList

Объект ObjectList является мобильным эквивалентом серверного элемента управления DataGrid ASP.NET. В частности, он может работать в двух режимах – списка и деталей. В режиме списка он выводит нечто вроде меню, составленного из значений одного поля. Так сделано в расчете на устройства с маленькими экранами. Свойство LabelField позволяет указать, какое связанное поле будет использоваться для заполнения этого меню. Пользователь может щелкать элементы, после чего производится переключение в режим деталей, в котором отображаются все поля текущей строки данных. Как обеспечивается возможность щелкать элементы, зависит от конкретного устройства.

В режиме деталей может отображаться панель инструментов с командами, заданными *с* использованием тэга <command>, и командой возврата, текст которой задается в свойстве BackCommandText. Наконец, свойство AutoGenerateField (по умолчанию устанавливаемое в *true)* совместно с элементами <fields> используется для выбора связанных полей, подлежащих отображению.

Элемент управления ObjectList имеет несколько отличий от элемента управления List. В частности, ObjectList позволяет связать с каждым элементом несколько команд. Набор команд может сов-

местно использоваться всеми элементами или быть у каждого элемента своим. Еще одно отличие заключается в том, что ObjectList поддерживает несколько представлений и не поддерживает статических элементов. Далее в этом разделе будет рассмотрена разработка демонстрационного приложения, в котором с помощью элемента управления ObjectList отображаются некоторые данные.

Элементы управления ObjectList и List полностью поддерживают внутреннюю разбивку на страницы и при выводе очередной части пользовательского интерфейса генерируют событие Loadltems. Разработчик задает общее количество элементов в свойстве ItemCount. По умолчанию это свойство содержит 0, и изменение его значения сигнализирует о том, что для элемента будет применяться пользовательская разбивка на страницы. Ожидается, что обработчик события LoadItems будет извлекать необходимые данные и связывать их с элементом управления.

Элемент управления SelectionList

Элемент SelectionList отличается от других списочных элементов управления для мобильных устройств тем, что позволяет выбирать пользовательский интерфейс списка элементов. Аналогично раскрывающемуся списку или набору переключателей он выводит список элементов и может предоставлять пользователю возможность выбирать из них один или несколько. При выборе элемента не происходит автоматическое генерирование серверного события, как это бывает с некоторыми другими списочными элементами управления, когда свойство AutoPostBack установлено в true. Когда выделяется другая группа элементов, генерируется событие SelectedIndexChanged.

Элемент управления SelectionList поддерживает пять разных вариантов рендеринга: раскрывающийся список (по умолчанию), набор кнопок-переключателей, набор флажков, обычный список, список с множественным выделением. Способ выбора (тип рендеринга) задается в свойстве SelectType. Не все устройства поддерживают все перечисленные способы выбора. Например, сотовые телефоны обычно поддерживают только списки с множественным выбором и флажки; переключатели, списки и раскрывающиеся списки превращаются в списки с множественным выбором.

Код на листинге 3.49, показывает, как отобразить список элементов и узнать, какие из них выбраны.

Листинг 3.49.

```
<mobile:Form runat="server" id="Form1">
   Your skills? <br />
   <mobile:SelectionList runat="server" id="list"
```

```
      SelectType="checkbox" />
   <mobile:Command runat="server" OnClick="Button_Click"
         Text="Go" />
</mobile:Form>
<mobile:Form runat="server" id="ResultsForm">
     <mobile:Label runat="server" id="Results" />
</mobile:Form>
```

Рис. 3.35. Элемент управления `SelectionList` *в действии*

Отделенный код этой формы заполняет элемент управления связанными данными (листинг 3.50).

Листинг 3.50.

```
public partial class Select:
     System.Web.UI.MobileControls.MobilePage.
{
   protected void Page_Load( object sender, EventArgs e)
   {
      if(!isPostBack) {
         ArrayList values = new ArrayList();
         values.Add( "ASP. NET");
         values.Add("ADO. NET");
         values.Add ("SQL Server");
         values.Add( "XML");
         list.DataSource = values;
         list.DataBind();
      }
   }
   protected void Button Click(object sender. EventArgs e)
   {
      string buf = "";
      foreach (MobileListItem item in list.Items)
         buf += (item.Selected ? item.Text + ", " : "");
```

```
        buf = buf.TrimEnd();
        if (buf.EndsWith(","))
           buf = buf.TrimEnd(',');
        Results.Text = buf;
        ActiveForm = ResultsForm;
    }
}
```

На рис. 3.35 данное приложение показано в действии. Слева виден заполненный элемент `SelectionList`, справа – результаты выбора, а посередине – командную кнопку.

3.4.5. Текстовые элементы управления

В ASP.NET определен абстрактный класс `TextControl`, наследуемый некоторыми навигационными элементами управления и элементами, служащими для ввода текста. Все текстовые элементы управления имеют свойство `Text`, которое можно устанавливать программно и которое обычно используется для рендеринга вывода элемента управления. Примерами элементов управления, производных от класса `TextControl`, являются `Link`, `TextBox` и `Label`. Мобильный элемент управления `Label` подобен одноименному классическому элементу управления ASP.NET, а мобильный элемент управления `Link` – одноименному классическому элементу управления `HyperLink`. Текстовыми элементами управления являются также `Command`, представляющий командную кнопку, и `PhoneCall`.

Элемент управления `TextBox`

Элемент управления `TextBox` генерирует однострочное текстовое поле и хранит в свойстве `Text` введенный пользователем текст. Данный элемент может работать в режиме пароля и в числовом режиме, но не поддерживает многострочного редактирования и не может выводиться как поле, доступное только для чтения. Можно задавать для него способ выравнивания и максимальную длину.

```
<mobile:TextBox runat="server" id="theUser"
     OnTextChanged="Alert" />
```

Элемент управления *TextBox* поддерживает серверное событие *OnTextChanged,* обработчик которого выполняется, когда между двумя последовательными операциями возврата формы значение текстового поля изменяется.

3.4.6. Элемент управления Command

Элемент управления `Command` обладает большинством характеристик классических элементов управления ASP.NET `Button` и `LinkButton`. Он может выводиться как кнопка типа `submit` (по умолчанию)

или гиперссылка и при этом быть представлен изображением. Свойство `Format` позволяет выбирать между типами `Button` и `Link`; если задан тип `Link`, элемент можно выводить в виде изображения, задав URL адрес этого изображения в свойстве `ImageUrl`. Если речь идет об устройстве, поддерживающем программируемые клавиши, элемент `Command` может выводить надпись для одной из таких кнопок. (Программируемые клавиши поддерживаются многими мобильными телефонами.)

Щелчок элемента управления `Command` вызывает событие возврата формы. Серверный код обработки этого события можно связать с событием `ItemCommand` или `OnClick`. Если используется событие `ItemCommand`, необходимо задать имя команды, а при желании и ее аргументы. Когда пользователь щелкнет элемент управления, по значению булева свойства `CausesValidation` станет явно, должен ли данный элемент управления выполнить проверку всех остальных элементов управления той же формы. Приведенный в листинге 3.51 код показывает, как использовать элементы управления `TextBox` и `Command`.

Листинг 3.51.

```
<mobile:Form runat="server"> Search for:
     <mobile:TextBox runat="server" font-bold="true"
     id="theSubject" />
     <mobile:Command runat="server" Text="GO"
          OnClick="OnSearch" />
</mobile:Form>
<mobile:Form runat="server" id="ResultsForm">
     <mobile:Label runat="server" id="theResults" />
</mobile:Form>
```

Рис. 3.36. Элементы управления `TextBox` и `Command` в действии

Когда пользователь щелкает командную кнопку, на сервере выполняется следующий код, показанный на листинге 3.52.

Листинг 3.52.

```
protected void OnSearch(object sender, EventArgs e)
{
     string msg = "Results for '{0}'";
     msg = String.Format(msg, theSubject.Text);
     theResults.Text = msg;
     ActiveForm = ResultsForm;
}
```

Свойство `ActiveForm` объекта `Form` позволяет программно выбирать текущую форму или узнавать, какая форма является текущей. Данное приложение в действии показано на рис. 3.36.

3.4.7. Элемент управления `PhoneCall`

Элемент управления `PhoneCall` используется для вывода на экран номера телефона (листинг 3.53). При этом, если устройство позволяет осуществлять звонки (как сотовый телефон), пользователь может щелкнуть этот номер телефона, чтобы по нему позвонить. В противном случае номер выводится как гиперссылка или даже просто как текст.

Листинг 3.53.

```
<mobile:Form runat="server">
     Phone numbers found for Joe:
     <mobile:PhoneCall runat="server"
          AlternateFormat="{0} at {1}"
          AlternateURL="http://www.acme.com"
          PhoneNumber="111-222-0000"
          Text="ACME Corp" />
     <mobile:PhoneCall runat="server"
          AlternateFormat="{0} at {1}"
          PhoneNumbe r="111-333-0000"
          Text="Home" />
</mobile:Form>
```

Свойство `AlternateFormat` может содержать любую строку, оно используется для формирования вывода на устройствах, не обладающих функцией телефонии. Данное свойство может принимать два параметра: в первый подставляется значение атрибута `Text`, а во второй – атрибута `PhoneNumber`. Альтернативный текст выводится как чистый текст, когда не задано свойство `AlternateURL`, превращающее его в гиперссылку на указанный в этом свойстве URL. Если атрибут `Text` не задан, на экран вы-

Рис. 3.37. Элемент управления `PhoneCall`

водится номер телефона. На рис. 3.37 показан вывод элемента управления PhoneCall в сотовом телефоне с браузером Openwave.

3.4.8. Проверочные элементы управления

Все проверочные мобильные элементы управления являются производными от абстрактного класса BaseValidator, который, в свою очередь, наследует базовый класс TextControl. Все элементы управления имеют свойства Text и ErrorMessage. Значение свойства Text выводится элементом управления в том случае, когда введено недопустимое значение, а если оно не задано, валидатор выводит значение свойства ErrorMessage. Свойство ErrorMessage имеет еще одно назначение – содержащийся в нем текст выводится в элементе управления ValidationSummary. Для мобильных элементов управления определены следующие пять валидаторов:

- **CompareValidator** – сравнивает значения двух элементов управления, используя заданный оператор сравнения. В свойстве ControlToValidate задается идентификатор проверяемого элемента управления, а значение, с которым производится сравнение, задается либо явно в свойстве ValueToCompare, либо неявно через свойство ControlToCompare. Последнее содержит идентификатор другого элемента управления, значение которого будет использоваться для сравнения.
- **CustomValidator** – позволяет проверить значение, используя собственный метод. В его свойстве ControlToValidate задается проверяемый элемент управления, а код проверки вставляется в обработчик серверного события ServerValidate. Для того чтобы проинформировать исполняющую среду о результатах проверки, нужно устанавливать свойство IsValid структуры данных, представляющей событие, в false или true.
- **RangeValidator** – выясняет, попадает ли значение заданного элемента управления в определенный диапазон. Границы диапазона задаются свойствами MinimumValue и MaximumValue, они должны быть указаны обе. В свойстве Type можно задать тип сравниваемых значений.
- **RegularExpressionValidator** – проверяет, соответствует ли значение заданного элемента управления определенному регулярному выражению. Проверяемый элемент управления задается в свойстве ControlToValidate, а выражение – в свойстве ValidationExpression.

- **`RequiredFieldValidator`** – производит контроль ввода для обязательного поля. При этом речь не всегда идет о том, что поле не должно быть пустым; в некоторых случаях важно, чтобы введенное значение отличалось от значения свойства `InitialValue` – только при этом условии проверка (выполняемая при потере полем фокуса ввода) считается успешной.

Когда в состав страницы включены валидаторы, ASP.NET выводит сообщения об ошибках проверки. Эта задача выполняется элементом управления `ValidationSummary`, который собирает все сообщения, заданные для разных валидаторов, объединяет их в одну строку и выводит результирующий текст во второй форме. Свойство `BackLabel` позволяет задать текст кнопки, служащей для возврата к исходной форме.

Элемент управления `Command` – единственный из мобильных элементов управления, вызывающий валидацию входных данных формы; `TextBox` и `SelectionList` – единственные элементы управления, значения которых подлежат валидации. У первого проверяется значение свойства `Text`, а у второго – значение свойства `Selectedlndex`. Однако и другие элементы управления (например, пользовательские и специализированные) могут участвовать в процессе валидации со своими собственными свойствами. Для этого их нужно пометить атрибутом *[*`ValidationProperty`*]*.

В листинге 3.54 показано, как создать страницу, проверяющую пользовательский ввод. Она содержит элемент управления `SelectionList`, в котором выводится перечень знаний кандидата на некоторую должность. Кандидат должен выбрать хотя бы один элемент в этом перечне. С элементом `SelectionList` связан валидатор `RequiredFieldValidator`. Значение свойства `InitialValue` ASP.NET автоматически устанавливает в –1.

Листинг 3.54.

```
<mobile:Form runat="server" id="Main">
     <mobile:RequiredFieldValidator runat="server"
          ErrorMessage="Must indicate a skill!"
          ControlToValidate="skills" />
     <b>Indicate your skills</b><br>
     <mobile:SelectionList runat="server" id="skills"
          SelectType="checkbox" />
     <mobile:Command runat="server" OnClick="Submit"
          Text="Send" />
</mobile:Form>
<mobile:Form runat="server" id="ResultsForm">
     <b>Recognized skills</b>
```

```
     <mobile:Label runat="server" id="Results" />
</mobile:Form>
```

Когда пользователь щелкает кнопку, служащую для передачи данных на сервер, страница активизирует все свои валидаторы, чтобы перед отправкой данных убедиться в их правильности (листинг 3.55).

Листинг 3.55.

```
protected void Page Load(object sender, EventArgs e)
{
     if (!isPostBack) {
          ArrayList values = new ArrayListO;
          values.Add("ASP.NET");
          values.Add("ADO.NET");
          values.Add("XML");
          values.Add("Windows Forms");
          values.Add("SQL Server");
          skills.DataSource = values;
          skills.DataBind();
     }
}
protected void Submit(object sender, EventArgs e) {
     if (!Page.IsValid) return;
     string buf = "";
     foreach (MobileListltem item in skills.Items)
          buf += (item.Selected ? item.Text + ", " : "");
     buf = buf.TrimEnd();
     if (buf.EndsWith(','))
          buf = buf.TrimEnd(',');
     Results.Text = buf;
     ActiveForm = ResultsForm;
}
```

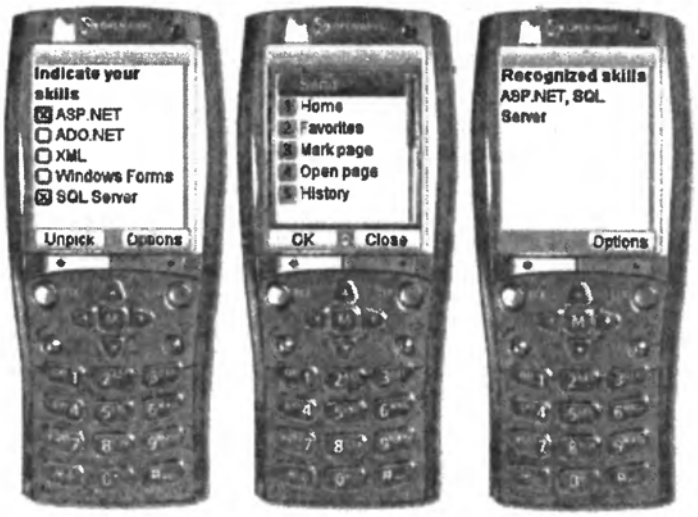

Рис. 3.38. Проверка пользовательского ввода с применением валидаторов

Если свойство `IsValid` объекта `MobilePage` содержит значение `false`, процедура завершает свою работу и клиенту возвращается та же форма с валидатором, отображающим сообщение об ошибке. При желании можно использовать вторую форму с элементом управления `ValidationSummary`, активизируемую по требованию. Действие приведенного выше кода показано на рис. 3.38.

3.4.9. Приложение для поиска клиентов

Для того, чтобы объединить все то, что пояснялось ранее об элементах управления и страницах мобильных приложений, поясним разработку небольшого демонстрационного приложения. В зависимости от возможностей целевого устройства разработчик приложения либо обеспечивает кэширование на клиентском устройстве той информации, которая затем будет передаваться серверной системе, либо реализует взаимодействие между клиентской и серверной системами в реальном времени. Приложения первого типа чаще создают для интеллектуальных устройств вроде Pocket PC, а второго – для устройств с малым объемом памяти и маломощными процессорами, таких как сотовые телефоны. Однако ограниченные возможности программного и аппаратного обеспечения подобных устройств затрудняют процесс создания эффективных и полезных приложений. Наиболее распространенным видом приложений для маломощных мобильных устройств являются программы для поиска информации. Такая программа и будет разрабатываться.

В рассматриваемом примере будет создаваться приложение, с помощью которого пользователи смогут запрашивать информацию о клиенте и, возможно, звонить ему по телефону. Безусловно, предмет поиска (клиент) выбран произвольно, с тем же успехом в большой базе данных можно искать любую другую информацию. Понятие *большой* базы данных здесь условно – она велика в сравнении с возможностями мобильного устройства. Если в обычном web-приложении можно выводить по 20 записей за раз, то в мобильном все, что можно предложить пользователю, – это несколько компактных элементов данных, помещающихся на маленьком экране.

При разработке мобильного приложения обязательно нужно учитывать и возможности ввода, поддерживаемые устройствами, для которых данное приложение предназначено. Если в мире Web пользователю часто предлагают ввести критерий, позволяющий сузить область поиска, то для беспроводного устройства такое решение подходит не всегда, поскольку выполнять текстовый ввод на нем неудобно.

Наше приложение будет извлекать информацию о клиенте, которого пользователь хотел бы посетить или с которым он хотел бы связаться

по телефону. При этом в памяти устройства база данных всех клиентов отсутствует – если это сотовый телефон, она туда просто не помещается, а если это более интеллектуальное устройство, скажем, КПК, то, может быть, пользователь забыл синхронизировать его базу данных с серверной или хочет получить самую свежую информацию.

Разработка архитектуры приложения

Многие настольные приложения выполняют поиск инкрементально – пользователь вводит частичный ключ и система возвращает список соответствий. Затем пользователь либо сужает область поиска, либо просто выбирает один из предложенных вариантов. Такой способ работы предполагает наличие удобной клавиатуры, но у мобильных устройств ее обычно не бывает. Как правило, такое устройство снабжено цифровой клавиатурой, но алфавитной – далеко не всегда. А рассчитывать на то, что пользователю понравится постоянно вводить текст, пользуясь цифровой клавиатурой, явно не приходится, особенно если учесть, что мобильные устройства потому и называются мобильными, что ими часто пользуются на ходу или в дороге, работая одной рукой.

Поэтому разработчик мобильного приложения, в особенности предназначенного для сотовых телефонов, должен хорошо усвоить следующее правило: один дополнительный щелчок предпочтительнее набора текста. Для персонализации и оптимизации вывода приложения используйте имеющуюся информацию о пользователе (его имя, адрес электронной почты, профиль или даже местоположение).

Структура формы должна быть как можно более простой и сжатой, с краткими, но понятными описаниями и единственным способом выполнения каждой задачи. Данные следует загружать малыми порциями, поскольку полоса пропускания приложения мала, а работа с приложением должна осуществляться путем прокрутки и нажатия программируемых клавиш, а также числовых клавиш на клавиатуре.

Для мобильных приложений, требующих пользовательского ввода, хорошо подходит интерфейс мастера. Очень удобно древовидное представление заданий и данных, поскольку оно позволяет предложить список вариантов, чтобы пользователь мог выбрать один из них или выйти. Имея все это в виду, перейдем к практике.

Навигационная модель приложения

Наше приложение `CustomerFinder` будет состоять из трех форм. Первые две служат для сужения области поиска клиента. Когда останется относительно небольшой их список, данные будут помещены в компонент `ObjectList` для более детального отображения. Таким образом,

пользователю нужно будет только нажимать кнопки, но не придется ничего вводить. Весь набор данных, в котором производится поиск, представлен в виде дерева, и при его обходе пользователь на каждом шаге осуществляет выбор дальнейшего пути. При этом на каждом шаге осуществляется загрузка очень маленького объема данных. Поскольку таблица клиентов используется во всех сеансах, дерево создается в памяти только раз и сохраняется в кэше ASP.NET для повышения производительности.

Главная форма

Первая форма приложения включает элемент управления `List`, отображающий четыре строки, которые вместе содержат 26 букв алфавита (листинг 3.56). Свойство `Text` объекта `ListItem` определяет диапазон, а свойство `Value` содержит буквы, разделенные запятыми.

Листинг 3.56.

```
<mobile:form id="MainForm" runat="server">
<mobile:List id="Menu" runat="server"
     OnItemCommand="Menu_ItemCommand">
     <Item Value="A,B,C,D,E,F" Text="A-F" />
     <Item Value="G,H,I,J,K,L,M" Text="G-M" />
     <Item Value="N,0,P,Q,R,S" Text="N-S" />
     <Item Value=T,U.V,W,X,Y,Z" Text="T-Z" />
</mobile:List>
<mobile:Label id="Desc" runat="server"
     Text="Select Customers" />
     <mobile:TextBox id="Initials" runat="server"
        Size="5" />
     <mobile:Command id="FindButton" runat="server"
     OnClick="FindButton_Click">Find</mobile:Command>
</mobile:form>
```

Внизу формы располагаются текстовое поле и командная кнопка. Когда пользователь щелкает эту кнопку, текст из поля ввода используется для выбора, отображаемого в следующей форме подмножества клиентов. При выборе одного из элементов списка выполняется код, показанный в листинге 3.57.

Листинг 3.57.

```
void Menu_ItemCommand(object sender, ListCommandEventArgs e)
{
     string[] menuItems =
          e.ListItem.Value.Split(", ".ToCharArray());
     Session["AvailableInitials"] = menultems;
     LetterList.DataSource = menuItems;
```

```
        LetterList.DataBind();
        // Переход ко второй форме
        ActiveForm = SecondMenuForm;
}
```

Разделенный запятыми список букв превращается в массив строк и временно сохраняется в кэше сеанса. Позднее он понадобится для проверки текста, который пользователь должен ввести в текстовое поле *Find* для поиска. Тот же массив строк используется для заполнения другого объекта *List.* В данном случае экземпляр элемента управления связывается с динамически сгенерированными данными, а не со статическими данными.

Для осуществления выбора пользователю нужно просто нажимать клавиши на цифровой клавиатуре. Однако если он предпочитает текстовый ввод, то может воспользоваться текстовым полем для более точного поиска. На рис. 3.39 показана последовательность выводимых экранов приложения при работе пользователя, которому не нравится вводить текст.

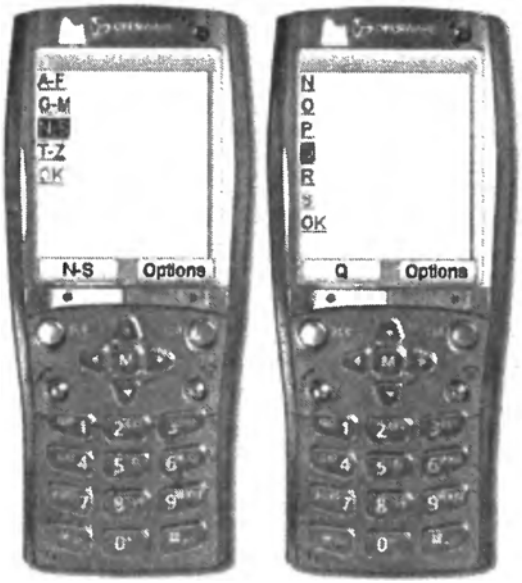

Рис. 3.39. Расширенная последовательность букв для сужения области поиска

Свойство `ActiveForm` используется в классе страницы для выбора и определения текущей активной формы. Когда страница вводится в первый раз, активизируется и отображается первая ее форма. После возврата этой формы на экране может появиться вторая форма, активизированная либо программно, либо в результате перехода, осуществленного пользователем. Свойство `ActiveForm` является одним из важнейших элементов, используемых при разработке интерфейса, который

строится по принципу мастера. В нашем примере после возврата формы свойству `ActiveForm` присваивается идентификатор второй формы-меню.

Вторая форма-меню

Следующая форма нашего приложения выводит буквы, перечисленные в свойстве `Value` выбранного элемента `ListItem`. Ее код приведен в листинге 3.58.

Листинг 3.58.

```
<mobile:form id="SecondMenuForm" runat="server">
     <mobile:List id="LetterList" runat="server" />
     <mobile:Label id="Desc" runat="server"
          Text="Select Customers" />
     <mobile:TextBox id="CustName" runat="server" />
     <mobile:Command id="FindCustomerButton" runat="server"
          text="Find" />
</mobile:form>
```

Когда пользователь щелкает одну из букв, код подготавливает в памяти запрос и возвращает объект ADO.NET `DataView`, содержащий список клиентов, имена которых начинаются с выбранной буквы. Этот список выводится с помощью элемента управления `ObjectList`.

Если пользователь хочет ввести строку, чтобы сузить область поиска, он может это сделать в текстовом поле, расположенном в нижней части второй формы (рис. 3.40). Текст, который здесь вводится, подлежит проверке – принимаются только те строки, которые начинаются с указанной буквы.

Рис. 3.40. Пользователь вводит текст, для более точной идентификации клиента

Форма со списком клиентов

Теперь наступила очередь вывода списка клиентов. Для этой цели используется объект `ObjectList`, который связан с набором строк, извлеченных кодом, показанном в листинге 3.59.

Листинг 3.59.

```
DataView SelectCustomers(string filterString)
{
     // Извлечение данных
     DataTable data = GetData();
     DataView view = new DataView(data);
     // Фильтрация данных
     string cmdText = "";
     string[] initials = filterString.Split
          (",".ToCharArray());
     string opOR = "";
     foreach(string s in initials) {
          cmdText += String.Format("{0} {2} LIKE '{1}%' ",
               opOR, s, "companyname");
          opOR = " OR ";
     }
     // Возврат отфильтрованных данных
     view.RowFilter = cmdText;
     return view;
}
```

Код связывает объект-представление с глобальной таблицей клиентов, загруженной в начале работы приложения. Этот объект с предварительно отфильтрованными данными присваивается свойству *DataSource* элемента управления `ObjectList` (листинг 3.60).

Листинг 3.60.

```
void LetterList_ItemCommand(object sender,
     ListCommandEventArgs e)
{
     DataView view = SelectCustomers(e.ListItem.Value);
     CustomerList.DataSource = view;
     CustomerList.DataBind();
     ActiveForm = CustomerForm;
}
```

Структура третьей формы показана в листинге 3.61.

Листинг 3.61.

```
<mobile:form id="CustomerForm" runat="server"
     paginate="true">
   <mobile:ObjectList id="CustomerList"
         runat="server" Wrapping="nowrap"
         AutoGenerateFields="False" LabelField="Company">
      <Field Name="Company" DataField="companyname"
```

```
          Visible="False" />
      <Field Name="Address" DataField="address" />
      <Field Name="City" DataField="city" />
      <Field Name="Country" DataField="country" />
      <Field Name="Phone" DataField="phone" />
   </mobile:ObjectList>
   <mobile:Command id="BackMenuCommand" runat="server"
     text="Back" />
</mobile:form>
```

В элементе управления `ObjectList` выводится список значений, связанных с ним посредством атрибута `LabelField`. Когда пользователь щелкает один из этих элементов, выводится детальная информация о выбранном клиенте (рис. 3.41) – ее состав определяется элементами разметки `<field>`. В число отображаемых полей должно входить и поле, заданное в атрибуте `LabelField`, в нашем примере – поле `Company`. Если нет желания, чтобы оно выводилось, установите его атрибут `Visible` в `false`. Атрибут `Name` позволяет определить псевдоним для поля исходной таблицы данных. Код третьей формы приведен в листинге 3.62.

Листинг 3.62.

```
<mobile:form id="CustomerForm" runat="server"
paginate="true">
   <mobile:ObjectList id="CustomerList" runat="server"
         Wrapping="nowrap" AutoGenerateFields="False"
         LabelField="Company">
      <Field Name="Company" DataField="companyname"
         Visible="False" />
      <Field Name="Address" DataField="address" />
      <Field Name="City" DataField="city" />
      <Field Name="Country" DataField="country" />
      <Field Name="Phone" DataField="phone" />
   </mobile:ObjectList>
   <mobile:Command id="BackMenuCommand" runat="server"
      text="Back" />
</mobile:form>
```

В элементе управления `ObjectList` выводится список значений, связанных с ним посредством атрибута `LabelField`. Когда пользователь щелкает один из этих элементов, выводится детальная информация о выбранном клиенте (рис. 3.57), ее состав определяется элементами разметки `<field>`. В число отображаемых полей должно входить и поле, заданное в атрибуте `LabelField`, в нашем примере – поле `Company`. Если не требуется, чтобы оно выводилось, то нужно задать его атрибуту

`Visible` значение `false`. Атрибут `Name` позволяет определить псевдоним для поля исходной таблицы данных.

Рис. 3.41. Итог работы приложения – найденная информация о клиенте

Представление, генерируемое элементом управления `ObjectList`, равно как и элементом управления `List`, может быть частично настроено с использованием шаблонов.

Вопросы для самопроверки

1. Какие типы мобильных приложений вы знаете?
2. Что такое эмулятор мобильного приложения, почему и зачем он используется?
3. Какие основные элементы управления используются в локальных мобильных приложениях?
4. Что такое web-сервис и как он может использоваться с мобильными приложениями?
5. Какие проблемы работы с web-сервисами вы знаете?
6. Что такое мобильное web-приложение?
7. Что такое адаптивный рендеринг?

ЗАКЛЮЧЕНИЕ

В связи с быстрым развитием мобильных вычислительных устройств и их популярности у пользователей будет постоянно увеличиваться и потребность в мобильных сервисах, которые расширяют возможности этих мобильных устройств. Однако разработка мобильных сервисов все еще остается сложной задачей, с которой могут справляться только программисты высокого уровня. Используемые мобильными приложениями коммуникационные средства предоставляют для них богатое разнообразие возможностей выбора, но это же обстоятельство повышает и ответственность разработчика за правильность сделанного им выбора с точки зрения соответствия этого выбора запросам пользователей мобильного приложения и требованиям экономичности. В силу доступности в настоящее время различных способов связи, при создании мобильных приложений открывается простор для самых смелых инноваций. Как и любая другая сфера разработки программного обеспечения для мобильных устройств, данная область широко открыты для исследований, и лучшим залогом удовлетворения коммуникационных потребностей новых мобильных приложений являются эксперименты и новаторство.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Волков В.Б. Программирование для карманных компьютеров. Самоучитель. – СПб.: Питер, 2006. – 304 с.
2. Горнаков С.Г. Программирование мобильных телефонов на Java 2 Micro Edition. – М.: ДМК Пресс, 2004. – 336 с.
3. Горнаков С.Г. Symbian OS. Программирование мобильных телефонов на C++ и Java 2 ME. – М.: ДМК Пресс, 2005. – 448 с.
4. Климов А.П. Программирование КПК и смартфонов на .NET Compact Framework. – СПб.: Питер, 2007. – 320 с.
5. Рихтер К., Охрин И. Мобильный бизнес: обзор, бизнес-приложения и перспективы // Журнал «Вестник Санкт-Петербургского университета». – 2005. – Выпуск 4. – С. 24–47.
6. Салмре И. Программирование мобильных устройств .NET Compact Framework / пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 736 с.
7. Эспозито Д. Microsoft ASP.NET 2.0 Базовый курс / пер. с англ. – М.: Издательство «Русская Редакция», 2007. – 688 с.
8. Эспозито Д. Microsoft ASP.NET 2.0 Углубленное изучение / пер. с англ. – М.: Издательство «Русская Редакция», 2007. – 592 с.
9. Mikkonen T. Programming mobile devices an introduction for practitioners. – London: John Wiley & Sons Ltd., 2007. – 245 p. (Simbian, Java)
10. Paavilainen J. Mobile business strategies – Understanding the technologies and opportunities. – London: IT Press, 2002. – 257 p.
11. Verbraeck A. Designing Mobile Service Systems. – Amsterdam: IOS Press, 2007. – 249 p.
12. Wigley A., Mothand D., Foot P. Microsoft Mobile Development Handbook. – NY: Microsoft Press, 2007. – 681 p.
13. Wigley A., Peter Roxburgh P. Building .NET Applications for Mobile Devices. – Redmond: Microsoft Press, 2002. – 657 p.
14. Windows Mobile 2003. – М.: МакЦентр–Бестселлер, 2005. – 236 с.
15. You P., Durant D. .Net Compact Framework Programming with C#. – NY: Printice Hall PTR, 2004. – 1424 p.

Учебное издание

ТУЗОВСКИЙ Фёдор Анатольевич

СЕТЕВЫЕ СЕРВИСЫ С ИСПОЛЬЗОВАНИЕМ КПК И СОТОВЫХ ТЕЛЕФОНОВ

Учебно-методическое пособие

Ассистент каф. ОСУ
Ф.А. Тузовский

Издано в авторской редакции

Компьютерная верстка *К.С. Чечельницкая*
Дизайн обложки *О.Ю. Аршинова*

Подписано к печати 20.08.2011. Формат 60x84/16. Бумага «Снегурочка».
Печать XEROX. Усл. печ. л. 8,72. Уч.-изд. л. 7,63.
Заказ ___-11. Тираж 35 экз.

Национальный исследовательский Томский политехнический университет
Система менеджмента качества
Издательства Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту BS EN ISO 9001:2008

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30
Тел./факс: 8(3822)56-35-35, www.tpu.ru