

Создание РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ на Java 2

- Графическая библиотека Swing
- Распределенные объекты RMI и CORBA
- Сервлеты, JSP и компоненты EJB
- Русификация распределенных приложений



Ильдар Хабибуллин

**Создание
РАСПРЕДЕЛЕННЫХ
ПРИЛОЖЕНИЙ
на Java 2**

Санкт-Петербург

«БХВ-Петербург»

2002

УДК 681.3.06
ББК 32.973.26-018.1
X12

Хабибуллин И. Ш.

X12 Создание распределенных приложений на Java 2. — СПб.: БХВ-Петербург, 2002. — 704 с.: ил.

ISBN 978-5-94157-106-2

Книга посвящена разработке многослойных приложений, отдельные части которых, возможно, распределены по сети, включая Internet, но работают как единое целое. В книге дано полное изложение стандартных средств Java 2 для создания распределенных приложений любой сложности, в том числе и Web-приложений. В качестве примера приведена система дистанционного обучения.

*Для разработчиков распределенных приложений,
студентов старших курсов и специалистов по информационным технологиям,
интересующихся новейшими средствами разработки*

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Анна Кузьмина</i>
Редактор	<i>Татьяна Коротяева</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Наталья Перишаква</i>
Дизайн обложки	<i>Игоря Цырульниковца</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.05.02.
Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 56,76.
Тираж 4000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 978-5-94157-106-2

© Хабибуллин И. Ш., 2002
© Оформление, издательство "БХВ-Петербург", 2002

Содержание

Введение	1
Структура книги.....	3
ЧАСТЬ I. ПОНЯТИЕ РАСПРЕДЕЛЕННОГО ПРИЛОЖЕНИЯ.....	7
Глава 1. Распределенные приложения и Java	9
Сетевые средства стандарта Java	9
Сокеты потокового типа TCP.....	13
Сокеты дейтаграммного типа UDP.....	15
Новые средства Java API.....	17
Механизм распределенных объектов RMI.....	18
Использование средств CORBA.....	22
Использование технологии WWW	27
Сервлеты	27
Java 2 Enterprise Edition.....	28
Сервер J2EE.....	29
Инсталляция J2EE SDK.....	29
Простейший сервлет.....	30
Установка сервлета в контейнер	32
Виртуальные каталоги	32
Работа сервлетов по протоколу HTTP.....	33
Язык JSP	35
Компоненты EJB.....	37
Система именования JNDI	41
Установка компонента EJB в контейнер.....	41
Распределенное приложение Java	43
Глава 2. Пример: система дистанционного обучения	46
Объектно-ориентированное проектирование	47
Design Patterns (паттерны проектирования).....	48
Пользователи СДО	49
Действия ученика.....	49
Регистрация.....	49
Учебный процесс.....	50
Оплата учебы	50
Действия учителя	50
Действия администратора	50
Графические интерфейсы	51

Начальная страница.....	51
Страницы ученика	51
Страница регистрации	51
Страница подтверждения регистрации.....	51
Страница учебной темы.....	51
Страница вопросов.....	52
Страница оценки.....	52
Страницы учителя.....	52
Начальная страница	52
Страница нового курса.....	52
Страница изменения курса	52
Утилита администратора	52
Функции и модули СДО	52
Структура СДО.....	53
Объекты СДО	53
Связи между объектами	54
Атрибуты объектов.....	55
Класс <i>Student</i>	55
Класс <i>StudentAccount</i>	57
Класс <i>Teacher</i>	58
Класс <i>Course</i>	60
Класс <i>Topic</i>	61
Класс <i>Task</i>	61
База данных SDO	63
Заключение	65

Часть II. Клиентская часть распределенного приложения Java..... 67

Глава 3. Оформление клиентского приложения компонентами Swing 69

Состав библиотеки Swing.....	70
Основные компоненты Swing.....	73
Класс <i>JComponent</i>	74
Схема MVC в компонентах Swing.....	75
Надпись <i>JLabel</i>	77
Кнопки.....	79
Кнопка <i>JButton</i>	82
Флажок <i>JToggleButton</i>	83
Флажок <i>JCheckBox</i>	84
Переключатель <i>JRadioButton</i>	84
Раскрывающийся список <i>JComboBox</i>	87
Список выбора <i>JList</i>	89
Визуализация элементов списков	90
Счетчик <i>JSpinner</i>	93
Линейка прокрутки <i>JScrollBar</i>	95
Ползунок <i>JSlider</i>	96
Индикатор <i>JProgressBar</i>	97
Дерево объектов <i>JTree</i>	98

Построение меню	103
Строка меню <i>JMenuBar</i>	103
Меню <i>JMenu</i>	103
Меню <i>JMenuItem</i>	105
Меню <i>JCheckBoxMenuItem</i>	106
Меню <i>JRadioButtonMenuItem</i>	106
Всплывающее меню <i>JPopupMenu</i>	107
Панель выбора цвета <i>JColorChooser</i>	108
Окно выбора файла <i>JFileChooser</i>	110
Фильтр файлов <i>FileFilter</i>	110
Как получить выбранный файл	112
Дополнительный компонент	113
Замена изображений	115
Русификация Swing	116
Глава 4. Создание форм с помощью текстовых компонентов	118
Компонент <i>JTextComponent</i>	118
Модель данных — документ	119
Строка символов <i>Segment</i>	119
Запись текста в документ	120
Атрибуты текста	120
Удаление текста из документа	121
Фильтрация документа	121
Внесение структуры в документ	122
События в документе	123
Реализации документа	123
Установка модели данных	124
Вид	124
Контроллер — редактор текста	126
Курсор	127
Ограничение перемещения курсора	127
Реализации редактора	129
Раскладка клавиатуры	129
Поле ввода <i>JTextField</i>	131
Поле ввода пароля <i>JPasswordField</i>	133
Редактор объектов <i>JFormattedTextField</i>	134
Область ввода <i>JTextArea</i>	135
Текстовый редактор <i>JEditorPane</i>	136
Редактор <i>JTextPane</i>	137
Глава 5. Помещение результата запроса в таблицу	138
Класс <i>JTable</i>	138
Модель данных таблицы	140
Модель ячеек таблицы	141
Свойства столбца таблицы <i>TableColumn</i>	147
Модель столбцов таблицы	147

Заголовки столбцов таблицы <i>JTableHeader</i>	147
Модель выделения ячеек	150
Визуализация ячеек таблицы	150
Редактор ячеек таблицы	155
Глава 6. Контейнеры	160
Панель <i>JPanel</i>	160
Панель прокрутки <i>JScrollPane</i>	161
Двойная панель <i>JSplitPane</i>	164
Панель со вкладками <i>JTabbedPane</i>	166
Линейная панель <i>Box</i>	168
Менеджер размещения <i>BoxLayout</i>	169
Компоненты-заполнители	169
Менеджер размещения <i>SpringLayout</i>	172
Размеры <i>Spring</i>	172
Промежутки <i>Constraints</i>	173
Размещение компонентов	174
Панель инструментальных кнопок <i>JToolBar</i>	176
Интерфейс <i>Action</i>	178
Слоеная панель <i>JLayeredPane</i>	180
Корневая панель <i>JRootPane</i>	181
Окно <i>JWindow</i>	183
Диалоговое окно <i>JDialog</i>	184
Окно верхнего уровня <i>JFrame</i>	186
Внутреннее окно <i>JInternalFrame</i>	187
Рабочий стол <i>JDesktopPane</i>	189
Стандартные диалоговые окна <i>JOptionPane</i>	191
Окно с индикатором <i>ProgressMonitor</i>	196
Апплет <i>JApplet</i>	198
Глава 7. Обработка событий в Swing	201
Событие изменения предка <i>AncestorEvent</i>	206
Событие изменения свойств <i>PropertyChangeEvent</i>	206
Управление колесиком мыши	208
Событие изменения состояния <i>ChangeEvent</i>	210
Глава 8. Оформление рамок	212
Пустая рамка <i>EmptyBorder</i>	214
Прямолинейная рамка <i>LineBorder</i>	215
Объемная рамка <i>BevelBorder</i>	215
Объемная рамка с закругленными краями <i>SoftBevelBorder</i>	216
Врезанная рамка <i>EtchedBorder</i>	216
Рамка с изображением <i>MatteBorder</i>	217
Рамка с надписью <i>TitledBorder</i>	218
Сдвоенная рамка <i>CompoundBorder</i>	221
Создание собственных рамок	223

Глава 9. Изменение внешнего вида компонента	231
Получение сведений о свойствах L&F	233
Задание стандартного стиля L&F	235
Дополнительные стили L&F.....	238
Смена всего стиля L&F.....	238
Замена отдельных свойств стиля L&F.....	241
Темы стиля Java L&F	244
Глава 10. Прочие свойства Swing	249
Индивидуальные свойства экземпляра компонента	249
Прокрутка содержимого компонента	250
Передача фокуса ввода.....	251
Перенос данных методом Drag and Drop.....	256
Временная задержка <i>Timer</i>	258
Интерфейс пользователя СДО	260
Часть III. Связь приложения Java с базами данных	271
Глава 11. Соединение приложения с базой данных	273
Загрузка драйверов JDBC	275
Соединение с базой данных	278
Использование JNDI для соединения.....	279
Интерфейс <i>DataSource</i>	279
Работа с источником данных.....	280
Пул соединений	282
Соединение для распределенной транзакции.....	284
Соединение <i>Connection</i>	285
Управление транзакцией.....	285
Уровни изоляции	286
Прочие методы.....	286
Свойства базы данных.....	287
Глава 12. Оформление запроса	290
Интерфейс <i>Statement</i>	290
Получение экземпляра <i>Statement</i>	291
Формирование и выполнение запроса	292
Изменение таблиц базы	299
Общий случай.....	300
Предварительно откомпилированные запросы	303
Параметры откомпилированного запроса.....	303
Выполнение откомпилированного запроса	304
Хранимые процедуры	306
Регистрация выходных параметров.....	307
Выполнение хранимой процедуры.....	307
Escape-синтаксис	308
Escape-синтаксис даты и времени	309

Escape-синтаксис внешних соединений.....	309
Escape-синтаксис экранирования	309
Сгенерированные ключи.....	310
Глава 13. Обработка результатов запроса.....	311
Интерфейс <i>ResultSet</i>	311
Расширенный просмотр выборки	312
Дополнительные свойства выборки.....	313
Интерфейс <i>ResultSetMetaData</i>	314
Обработка выборки.....	315
Обновляемые результаты	316
Удаление строки.....	318
Вставка строки	318
Выборка типа <i>RowSet</i>	319
Соединение с источником данных	320
Заполнение выборки.....	324
Обработка результатов.....	325
Обработка событий.....	330
Глава 14. Прочие свойства интерфейса JDBC	332
Пакетное изменение источника данных	332
Типы данных <i>BLOB</i> и <i>CLOB</i>	334
Интерфейсы <i>Blob</i> и <i>Clob</i>	334
Получение большого объекта	336
Занесение большого объекта в источник данных	338
Составные типы данных	338
Интерфейс <i>Array</i>	340
Интерфейс <i>Struct</i>	340
Ссылочный тип данных <i>REF</i>	340
Типы данных пользователя, UDT.....	341
Тип данных <i>DISTINCT</i>	343
Двухслойная схема СДО	343
Часть IV. Сетевая часть приложения Java	353
Глава 15. Обращение к удаленным методам.....	355
Структура системы RMI	356
Создание удаленного объекта.....	358
Удаленный интерфейс.....	358
Удаленный объект.....	359
Создание RMI-сервера.....	361
Создание RMI-клиента	362
Компиляция и запуск системы RMI	363
Аргументы и результат удаленного метода	368
Протоколы системы RMI	369
Протокол JRMP	370
RMI поверх HTTP.....	371

Активизация удаленных объектов.....	371
Группа активизации.....	375
Применение собственных сокетов.....	376
Использование RMI-реестра.....	380
Обратный вызов.....	381
Глава 16. Обмен сообщениями между частями распределенного приложения.....	387
Сообщение <i>Message</i>	388
Заголовок сообщения.....	389
Свойства сообщения.....	389
Тело сообщения.....	391
Соединение <i>Connection</i>	392
Сведения о соединении.....	394
Сеанс <i>Session</i>	394
Отбор сообщений.....	397
Адресат <i>Destination</i>	398
Отправители и получатели.....	399
Обмен сообщениями в СДО.....	402
Глава 17. Поиск распределенных объектов в сети.....	409
Имена.....	411
Имя класса <i>CompoundName</i>	412
Имя класса <i>CompositeName</i>	414
Имя в виде строки URL.....	416
Контекст.....	417
Начальный контекст <i>InitialContext</i>	418
Контекст каталога.....	420
Интерфейс <i>Attribute</i>	420
Класс <i>BasicAttribute</i>	421
Интерфейс <i>Attributes</i>	422
Класс <i>BasicAttributes</i>	422
Действия с атрибутами.....	423
Поиск объектов в каталоге контекста.....	425
Реализация контекста каталога.....	426
Ссылки на объекты.....	434
Поставщик услуг RMI.....	435
Поставщик услуг файловой системы.....	436
Поставщик услуг DNS.....	438
Поставщик услуг NIS.....	439
Поставщик услуг LDAP.....	440
Поставщик услуг COS Naming.....	441
Глава 18. Средства электронной почты.....	443
Замечания о кодировке.....	444
Письмо <i>Message</i>	447

Сеанс связи <i>Session</i>	450
Адрес <i>Address</i>	452
Связь <i>Service</i>	453
Хранилище <i>Store</i>	459
Получение писем	460
Пометки писем.....	462
Работа с присоединенной информацией	463
Отправка HTML-файла.....	466
Глава 19. Связь компонентов Java с архитектурой CORBA	469
Использование RMI-ИОП.....	471
Адаптер BOA.....	473
Использование COS Naming	476
Адаптер POA.....	478
Использование IDL	483
Долговременные объекты	488
Заключение	489
Часть V. Серверная часть распределенного приложения Java	491
Глава 20. Сервлеты и JSP	493
Интерфейс <i>Servlet</i>	494
Конфигурационный файл	495
Интерфейс <i>ServletConfig</i>	498
Контекст сервлета.....	501
Метод <i>Service</i>	501
Интерфейс <i>ServletRequest</i>	502
Интерфейс <i>ServletResponse</i>	503
Цикл работы сервлета	504
Класс <i>GenericServlet</i>	505
Работа по протоколу HTTP	506
Интерфейс <i>HttpServletRequest</i>	506
Интерфейс <i>HttpServletResponse</i>	508
Класс <i>HttpServlet</i>	509
Сеанс связи с сервлетом	516
Фильтры	520
Обращение к другим ресурсам	525
Страницы JSP	526
Теги JSP	530
Встроенные объекты JSP	533
Обращение к компоненту JavaBean	535
Выполнение апплета в браузере клиента	536
Передача управления.....	537
Заказные теги	538
Класс-обработчик заказного тега.....	541
Заказной тег с атрибутами	543

Заказной тег с телом.....	544
Обработка тела заказного тега.....	546
Обработка взаимодействующих тегов.....	549
Обработка исключений в заказных тегах.....	554
Стандартная библиотека тегов JSTL.....	555
Глава 21. Компоненты на сервере.....	557
Роли создателей компонентов EJB.....	559
Интерфейсы компонентов EJB.....	560
Связь с клиентом.....	561
Состав компонента EJB.....	563
Разновидности компонентов EJB.....	568
Контекст компонента EJB.....	569
Метаданные компонента.....	570
Идентификаторы компонента EJB.....	571
Клиентские заглушки.....	572
Переменные окружения компонента.....	573
Ограничения компонентов EJB.....	576
Глава 22. Создание Session и MessageDriven Beans.....	577
Создание и удаление session-компонента.....	577
Активизация SFSB-компонента.....	581
Пример SFSB-компонента.....	581
Клиенты компонента EJB.....	589
Отдельное приложение.....	590
Сервлет.....	591
Страница JSP.....	594
Другой компонент EJB.....	596
Клиент, связанный с приложением J2EE.....	596
Компоненты, управляемые сообщениями.....	597
Глава 23. Создание Entity Beans.....	605
Home-методы.....	606
Интерфейс <i>EntityBean</i>	607
Методы создания экземпляра.....	609
Методы поиска.....	612
Состояния экземпляра entity-компонента.....	614
CMP-компонент.....	615
Отношения, управляемые контейнером.....	621
Методы выбора.....	623
Язык запросов EJB QL.....	623
CMP-компонент.....	626
Глава 24. Транзакции.....	637
Стандарты распределенных транзакций.....	639
Интерфейсы управления транзакцией.....	640

Прикладной интерфейс.....	642
Интерфейсы управления ресурсами.....	643
Транзакции в апплетах и приложениях.....	645
Транзакции в Web-компонентах.....	645
Транзакции в EJB-компонентах.....	647
Транзакции, управляемые компонентом.....	647
Транзакции, управляемые контейнером.....	653
Глава 25. Технология Java и язык XML.....	655
Пространство имен тегов.....	659
Анализ документа XML.....	660
Анализ документов XML с помощью SAX2 API.....	661
Анализ документов XML с помощью DOM API.....	664
Интерфейс <i>Node</i>	665
Интерфейс <i>Document</i>	667
Интерфейс <i>Element</i>	668
Преобразование дерева объектов в XML.....	671
Таблицы стилей XSL.....	674
Преобразование документа XML в HTML.....	676
Глава 26. Упаковка и установка компонентов.....	678
Утилита упаковки.....	683
Утилита проверки.....	685
Утилита установки.....	685
Список литературы и Internet-источников информации.....	687
Предметный указатель.....	689

Введение

Сразу после выхода книги [15] я стал получать много отзывов о ней. Среди критики и указаний на многочисленные опечатки оказалось удивительно много предложений расширить приложение к книге, названное "Развитие Java", и подробно описать затронутые в нем вопросы, относящиеся к серверной части технологии Java.

Это непростая задача.

Технология Java быстро превратилась в мощную и развитую платформу для построения крупных информационных систем уровня предприятия. В ней есть все средства для создания рабочих мест клиента, оснащенных полноценным графическим интерфейсом, и серверов приложений, предоставляющих все необходимые клиентам услуги, а также для установления сетевой связи между различными удаленными частями информационной системы.

Воспользоваться средствами Java очень легко. Все необходимое для разработки распределенного приложения Java находится в двух самораспаковываемых архивах, которые можно свободно загрузить с сайта фирмы Sun Microsystems. Они расположены по адресам <http://java.sun.com/j2se/> и <http://java.sun.com/j2ee/>. Надо только выбрать файл, соответствующий вашей операционной системе: MS Windows, Solaris, Linux, MacOS X. Я настоятельно рекомендую загрузить, кроме того, файл с документацией. Документация в формате HTML едина для всех операционных систем, поэтому она поставляется отдельным файлом.

Развернув первый архив на своем компьютере, вы получите набор J2SE SDK (Java 2 Standard Edition Software Development Kit), содержащий компилятор, интерпретатор, отладчик, набор базовых классов. Из второго архива вы получите J2EE SDK (Java 2 Enterprise Edition Software Development Kit), содержащий дополнительный набор классов для создания серверной части приложения, а также небольшой сервер приложений и утилиты для работы с ним. Остается только создать переменные окружения `JAVA_HOME`, `J2EE_HOME`, занести пути к исполняемым файлам в переменную `PATH`, как указано в прилагаемой инструкции по инсталляции, и можно работать.

Разумеется, лучше установить последние версии SDK. Ко времени написания книги это были версии J2SE SDK 1.4.0 и J2EE SDK 1.3.1.

Вы можете работать непосредственно с компилятором, отладчиком и интерпретатором из командной строки, но большинство разработчиков, особенно для совместной работы, выбирают какое-либо из многочисленных RAD-

средств разработки (Rapid Application Development). В России наиболее популярны следующие RAD-продукты:

- JBuilder фирмы Borland Software Corporation, <http://www.borland.com/>;
- Visual Age for Java фирмы IBM, <http://www.ibm.com/java/>;
- Kawa фирмы Allaire Corporation, <http://www.allaire.com/>;
- IDEA фирмы IntelliJ Software, <http://www.intellij.com/>;
- Together фирмы TogetherSoft Corporation, <http://www.togethersoft.com/>;
- Rational Rose фирмы Rational Software Corporation, <http://www.rational.com/>;
- если ваши данные хранятся в базе Oracle, то удобно использовать JDeveloper корпорации Oracle, <http://www.oracle.com/>.

Хотя средства разработки Java относительно компактны, сама технология Java настолько обширна, что книги, описывающие каждую ее часть, содержат по тысяче страниц. Для того чтобы книга, которую вы держите в руках, имела разумный объем, пришлось, во-первых, предположить, что читатель уже знаком со стандартными средствами Java, например, по книгам [1, 11, 15, 18], и с объектно-ориентированным программированием [3].

Во-вторых, пришлось ограничить книгу описанием средств создания только одного распределенного приложения, а не целой информационной системы. Особенности создания разветвленных систем уровня предприятия посвящена книга [12].

Под распределенным приложением в этой книге понимается информационная система, компоненты которой работают на различных машинах, часто удаленных друг от друга на значительное расстояние. Компоненты могут функционировать в разных операционных средах, зачастую построенных по совершенно непохожим принципам. Для связи элементов распределенного приложения могут применяться самые различные сетевые средства, но в этих сложных условиях распределенное приложение должно выполняться как единое целое, как будто все его компоненты работают на одной машине.

В-третьих, книга адресована профессиональному разработчику. Это не ограничивает круг читателей, поскольку в России пока только профессионалы интересуются технологией Java, в отличие от США, где это сейчас повсеместно распространенная технология создания распределенных систем. Я называю профессионалом того, кто, кроме всего прочего:

- свободно читает по-английски технические тексты;
- обязательно загрузит с адресов <http://java.sun.com/docs/books/tutorial/> и <http://java.sun.com/j2ee/> и установит на своем компьютере не входящие в документацию учебники Java Tutorial и J2EE Tutorial;
- просмотрит сайт <http://java.sun.com/blueprints/>, содержащий пример разработки распределенного приложения Java Pet Store и относящиеся к нему документы;

- ❑ познакомится со всеми спецификациями, описывающими отдельные продукты Java;
- ❑ внимательно прочтает документы и исходные тексты примеров из каталога \$JAVA_HOME/docs/guide/ и просмотрит демонстрационные материалы из каталога \$JAVA_HOME/demo;
- ❑ распакует архив \$JAVA_HOME/src.zip и будет обращаться к исходным текстам классов Java, чтобы узнать, как они работают;
- ❑ будет постоянно просматривать материалы, размещенные на сайтах <http://java.sun.com/>, <http://www.ibm.com/java/>, <http://www.theserverside.com/>, <http://www.javable.com/> и других многочисленных сайтах, посвященных технологии Java.

Эти профессиональные качества читателя позволяют уменьшить объем книги до разумного предела. Книга не содержит полных описаний классов Java. Они приведены в документации. В книге минимум схем и рисунков, потому что лучше, чем в Java Tutorial, их не сделаешь. В книге нет описания работы с графическими утилитами и RAD-продуктами — профессионал осваивает их по встроенной в продукт справке (Help). В книге нет пространных рассуждений, оценок и сравнений продуктов — у каждого профессионала есть свое твердое мнение на этот счет.

Что же есть в книге?

Книга представляет собой полное руководство по использованию Java для создания распределенных приложений. В ней описано ЧТО, какие средства, есть в технологии Java для создания распределенных приложений, и КАК ими пользоваться в повседневной практической работе. Подробно рассматриваются стандартные классы, из которых составляется приложение, и методы, которыми они обеспечивают работу приложения. Рассказано, как применять эти классы и методы для решения своих задач. Приведены примеры "из жизни", показывающие работу классов Java и их методов в распределенных приложениях. На протяжении всей книги разрабатывается гипотетическая распределенная система дистанционного обучения СДО.

Структура книги

Книга состоит из двадцати шести глав, сгруппированных в пять частей.

Часть I содержит две главы.

В *главе 1* рассматриваются базовые понятия распределенных приложений, их строение, приемы и средства Java для их разработки. Приведен краткий обзор стандартных пакетов классов и интерфейсов Java, необходимых для разработки распределенных приложений, даны примеры их использования. Эта глава поможет составить общее представление о структуре распределенного приложения Java.

В *главе 2* рассказывается о современных средствах проектирования и разработки сложных приложений. Они использованы для проектирования сквозного примера книги — системы дистанционного обучения, СДО. Рассмотрена многослойная структура распределенного приложения.

В *части II* подробно описывается клиентская часть распределенного приложения и основное средство для ее создания — графическая библиотека Swing. Для знакомства с ней потребовалось восемь глав.

В *главе 3* рассматриваются основные графические компоненты библиотеки Swing: метки, кнопки, списки, ползунки, меню, деревья и другие компоненты, их возможности и правила применения.

В *главе 4* описаны текстовые компоненты Swing и работа с текстом с их помощью: редактирование текста, замена цвета и шрифта, вставка компонентов в текст.

Создание таблиц и работа с ними объясняются в *главе 5*. Описаны приемы заполнения ячеек таблицы, их оформления цветом и шрифтом, выделение заголовков, отдельных строк и столбцов.

В *главе 6* показано, как разместить компоненты в контейнерах, какие виды контейнеров есть в библиотеке Swing и как их использовать для придания графическому интерфейсу пользователя удобного и красивого вида.

О том, как обработать события, вызываемые действиями пользователя, рассказывается в *главе 7*. Библиотека Swing содержит удобные и гибкие средства реагирования на подобные события, они подробно разобраны в этой главе.

Глава 8 посвящена рассмотрению различного вида рамок, обрамляющих графические компоненты и контейнеры.

Библиотека Swing предоставляет интересную возможность изменять внешний вид графического интерфейса в зависимости от платформы, на которой работает графическое приложение Java или, наоборот, сделать внешний вид приложения независимым от платформы. Эта возможность описана в *главе 9*.

Наконец, *глава 10* посвящена прочим возможностям библиотеки Swing, не описанным в предыдущих главах.

В *части III*, состоящей из четырех глав, рассматриваются средства Java для связи с базами данных.

В *главе 11* рассказывается о том, что такое JDBC, как выбрать драйвер JDBC, загрузить его и установить соединение с базой данных.

В *главе 12* рассматриваются различные виды запросов к базе данных: выборки, включение строк в таблицы, их изменение и удаление, создание таблиц и прочие действия с базами данных. В этой главе показано, как оформить и выполнить запрос средствами JDBC.

О том, как обработать результаты запроса, в частности, поместить их в ячейки таблицы Swing, рассказано в *главе 13*.

Глава 14 посвящена рассмотрению прочих свойств JDBC, не освещенных в предыдущих главах, в частности, работе с курсорами.

В *IV части* книги, состоящей из пяти глав, описаны сетевые средства Java, составляющие сердцевину всякого распределенного приложения.

В *главе 15* подробно объясняется механизм RMI и его использование для обращения к методам распределенных объектов, а также применение ПОР — транспортного протокола CORBA.

В *главе 16* рассмотрена система асинхронного обмена сообщениями JMS — реализация MOM средствами Java. Показано, как организовать очереди сообщений и разделы подписки, прием и отправку сообщений.

В *главе 17* рассказывается о средствах JNDI именования и поиска распределенных объектов в сети — распределенной службе каталогов Java.

В *главе 18* показано, каким образом с помощью почтовой службы JavaMail распределенное приложение можно включить во всемирную систему электронной почты, принимать и отправлять почтовые сообщения.

Наконец, в *главе 19* рассматривается язык Java IDL — средство включения объектов Java в архитектуру CORBA.

V часть книги посвящена серверной части распределенного приложения. Она состоит из семи глав.

В *главе 20* рассказывается о сервлетах и страницах JSP — основных средствах Java, предназначенных для расширения возможностей Web-сервера. С их помощью можно легко включить в Web-сервер дополнительные средства обработки информации.

В *главах 21, 22 и 23* подробно описываются компоненты EJB — переносимые компоненты, работающие на сервере и легко встраиваемые в любые J2EE-серверы приложений.

В *главе 24* показано, как можно средствами JTA объединить действия по обработке информации в транзакцию, управлять транзакциями и включать в транзакцию компоненты распределенного приложения и источники данных.

Все большее распространение получает язык XML. В *главе 25* изучаются средства пакета JAXP для обработки документов XML.

В *главе 26*, последней главе книги, описывается процедура упаковки компонентов распределенного приложения для переноса их на другой сервер и процедура установки компонентов на другом J2EE-сервере.



ЧАСТЬ I

ПОНЯТИЕ РАСПРЕДЕЛЕННОГО ПРИЛОЖЕНИЯ

Глава 1



Распределенные приложения и Java

В этой главе дан обзор средств, которые предлагает технология Java для создания распределенных приложений, и которыми мы будем пользоваться на протяжении всей книги. Для того чтобы читатель сразу получил представление о том, как реализуются эти средства, приведены листинги простейших примеров их использования.

Сетевые средства стандарта Java

Фирма Sun Microsystems, девиз которой: "Сеть — это компьютер", активно развивает сетевые технологии, стремясь в то же время упростить создание распределенных приложений. (К слову, у фирмы Sun с некоторых пор появился новый лозунг: "We are the dot in .com". Фирма Sun хочет сказать, что она полностью обеспечивает инструментами Internet-бизнес и ставит точку в умножении их числа.) Основную роль в упрощении разработки сетевых приложений фирма Sun Microsystems отводит технологии Java.

Уже в первый выпуск инструментального набора классов и утилит JDK 1.0 был включен пакет `java.net`, содержащий набор классов для связи отдельных частей приложения, работающих на разных машинах, по протоколу TCP посредством сокетов, для пересылки дейтаграмм UDP и для обмена информацией по протоколам, работающим посредством адресации URL. Эти классы предельно автоматизировали установку соединения клиента с сервером. Достаточно написать пять-семь строк кода на языке Java, и программа готова. Например, если отвлечься от проверок и обработки исключений, то, для того, чтобы получить домашнюю страничку сайта и отправить ее в стандартный вывод программы, необходимо написать только:

```
URL bhv = new URL("http://www.bhv.ru");
BufferedReader br = new BufferedReader(
    new InputStreamReader(bhv.openStream()));
String line = null;
while ((line = br.readLine()) != null)
    System.out.println(line);
```

Конечно, упрощение программирования, активно продвигаемое фирмой Sun, достигается за счет инкапсуляции всей сложности сетевого взаимодействия в классы Java. Такое "крупноблочное" программирование требует четкого взаимодействия классов на всех уровнях сетевых протоколов. Малейшее расхождение в реализации протокола приведет к сбоям в работе сетевой программы, а настроить ее невозможно, все параметры реализации протокола скрыты внутри классов. Можно только заменить один блок программы другим блоком. Много времени и сил уходит на подгонку программ разных производителей, подбор совместимых версий, наложение заплаток-патчей. Но все-таки мы перешли от программирования в машинных кодах к языку Ассемблера, а от него к языкам высокого уровня. Сейчас мы манипулируем целыми объектами и уже внедряем компонентное программирование, одним оператором объектно-ориентированного языка программирования вставляя в свое приложение электронную таблицу или окно открытия файла. Тенденцию к укрупнению "строительных блоков" не остановить, и фирма Sun только следует ей.

Впрочем, вместо класса URL можно применить класс URLConnection или его расширение, введенное в JDK 1.1, — класс HttpURLConnection, использующий особенности протокола HTTP. Методы этих классов позволяют задать параметры соединения, например, MIME-тип сообщения, его кодировку, метод передачи сообщения GET или POST, определить заголовок сообщения. При получении ответа от Web-сервера можно просмотреть код ответа, заголовок ответа, узнать MIME-тип ответа и его кодировку. Все это позволяет отрегулировать приложение, настроить взаимодействие клиентской и серверной частей программы.

В листинге 1.1 показано, как можно послать параметры *CGI-программе* (CGI — Common Gateway Interface, общедоступный интерфейс шлюза) методом POST протокола HTTP, и получить от нее ответ. Для правильной передачи символов кириллицы строка параметров, записанная в программе в кодировке Unicode, перед отправкой по сети преобразуется в байтовую кодировку KOI8-R.

Листинг 1.1. Посылка параметров CGI-программе

```
import java.net.*;
import java.io.*;

public class UsingCGI{
    public static void main(String[] args){
        String req = "Параметры, посылаемые CGI-программе";
        try{
            // Преобразуем строку Unicode в массив
```

```
// байтовых символов в кодировке KOI8-R.
byte[] data = req.getBytes("KOI8-R");

URL url =
    new URL("http://www.bhv.ru/cgi-bin/javastart.bat");

// Получаем экземпляр класса URLConnection.
URLConnection uc = url.openConnection();

// Устанавливаем свойства соединения
uc.setDoOutput(true); // Будем отправлять
uc.setDoInput(true); // и получать информацию.
uc.setUseCaches(false); // Кэш нам не нужен.
// Определяем MIME-тип и подтип посылаемого сообщения.
uc.setRequestProperty("content-type",
    "application/octet-stream");
// Для метода POST нужна длина сообщения.
uc.setRequestProperty("content-length",
    ""+req.length());

// Устанавливаем соединение.
uc.connect();

// Открываем выходной поток
DataOutputStream dos =
    new DataOutputStream(uc.getOutputStream());

// и выводим в него массив параметров.
dos.write(data, 0, data.length);

dos.close();

// Открываем входной поток, предполагая, что
// он записан в кодировке KOI8-R.
BufferedReader br = new BufferedReader(
    new InputStreamReader(uc.getInputStream(), "KOI8-R"));

String res = null;
// Читаем ответ строка за строкой.
```

```
while ((res = br.readLine()) != null)
    System.out.println(res);

br.close();

} catch (Exception e) {}
}
}
```

CGI-программу тоже можно написать на языке Java. В листинге 1.2 приведен пример простой CGI-программы.

Листинг 1.2. CGI-программа, написанная на языке Java

```
import java.io.*;

public class JavaCGI{
    public static void main(String[] args){
        try{
            BufferedReader br = new BufferedReader(
                new InputStreamReader(System.in, "KOI8-R"));

            String s = br.readLine();

            // Здесь обработка полученной строки параметров...

            PrintWriter pr = new PrintWriter(
                new OutputStreamWriter(System.out, "KOI8-R"), true);
            pr.println("Ответ: " + s);

        } catch (Exception e) {}
    }
}
```

Эта программа просто отправляет обратно первую строку полученного сообщения. Для правильной передачи символов кириллицы стандартный ввод и вывод программы принимают и посылают поток байтов в кодировке KOI8-R.

Файл `JavaCGI.class`, полученный после компиляции программы, приведенной в листинге 1.2, надо поместить в каталог `sgi-bin` или какой-то другой

каталог, в котором хранятся CGI-программы. Расположение этого каталога зависит от используемого Web-сервера. Кроме того, надо написать командный файл, содержащий всего одну строку:

```
java JavaCGI
```

и поместить его в тот же каталог. В листинге 1.1 этот командный файл назван `javastart.bat`. Именно он указан в URL-адресе CGI-программы. Web-сервер запустит соответствующий интерпретатор командной оболочки (shell), который, выполняя командный файл, запустит *виртуальную машину Java* (JVM — Java Virtual Machine). Она обратится к методу `main()` класса `JavaCGI`, передав на стандартный ввод программы полученную по сети строку параметров. После того как программа направит результаты своей работы в стандартный вывод, Web-сервер извлечет их оттуда и отправит клиенту.

Конечно, в производственной системе запускать новый экземпляр JVM при каждом обращении к CGI-программе нельзя. Это приведет к громадному расходу ресурсов. JVM надо запускать один раз при первом запросе клиента, а затем для обработки запросов создавать подпроцессы. Для реализации такого подхода в технологии Java разработана спецификация *сервлетов* (servlets) и реализованы специальные программы (servlets engine), запускающие сервлеты и следящие за их выполнением.

В состав пакета `java.net` включены классы, которые позволяют спуститься с прикладного уровня сетевого соединения на транспортный уровень и организовать соединения с помощью сокетов дейтаграммного типа по протоколу UDP или сокетов потокового типа по протоколу TCP.

Сокеты потокового типа TCP

Вся сложность установления связи по протоколу TCP спрятана в двух классах: `SocketServer` и `Socket`. На стороне клиента создается *сокет* — объект класса `Socket`. При создании объекта в него заносится адрес и номер порта сервера. На стороне сервера работает экземпляр класса `SocketServer`, "прослушивающий" назначенный ему порт. Сокет клиента устанавливает соединение с сервером и открывает входной и выходной потоки методами `getInputStream()` и `getOutputStream()` для обмена информацией с сервером. В листинге 1.3 показана схема работы клиента.

Листинг 1.3. Схема TCP-клиента

```
import java.net.*;
import java.io.*;

class TCPClient{
    public static void main(String[] args){
```

```
try{
    Socket sock = new Socket("www.bhv.ru", 1666);

    PrintWriter pw = new PrintWriter(
        new OutputStreamWriter(sock.getOutputStream()), true);

    String req = "Запрос клиента";
    pw.println(req);

    BufferedReader br = new BufferedReader(
        new InputStreamReader(sock.getInputStream()));

    String res = null;
    while ((res = br.readLine()) != null)
        System.out.println(res);

    // Обмен продолжается...

    // По окончании работы потоки и сокет закрываются.
    pw.close(); br.close();
    sock.close();

} catch(Exception e){}
}
```

На серверной машине сервер — экземпляр класса `ServerSocket` — ждет запрос от сокета клиента, слушая назначенный ему порт. Получив запрос, он соединяется с клиентом методом `accept()`, создав для связи свой сокет — экземпляр класса `Socket`. После этого сервер продолжает слушать порт, обратившись снова к методу `accept()`, и ждет запрос от другого клиента. Сокет сервера открывает входной и выходной потоки для обмена информацией с сокетом клиента. По окончании работы сокет закрывается. Листинг 1.4 показывает схему работы сервера.

Листинг 1.4. Схема TCP-сервера

```
import java.net.*;
import java.io.*;
```

```
class TCPServer{
    public static void main(String[] args){
        try{
            ServerSocket ss = new ServerSocket(1666); // 1666 – номер порта.

            while (true){
                Socket cl = ss.accept();

                // Для работы сокета cl лучше создать новый подпроцесс.

                BufferedReader br = new BufferedReader(
                    new InputStreamReader(cl.getInputStream()));
                String req = br.readLine();

                // Здесь обработка запроса.

                PrintWriter pw = new PrintWriter(
                    new OutputStreamWriter(cl.getOutputStream()), true);
                pw.println("Ответ: " + req);

                // Обмен данными продолжается...

                // По окончании обмена потоки и сокет закрываются.
                br.close(); pw.close(); cl.close();

                // Сервер продолжает слушать порт...
            }

        }catch(Exception e){}
    }
}
```

Классы `Socket` и `ServerSocket` содержат большое число методов, с помощью которых можно задать и получить различные параметры сокетов для настройки характеристик связи.

Сокеты дейтаграммного типа UDP

Обмен дейтаграммами с использованием классов пакета `java.net` происходит через сокеты дейтаграммного типа — экземпляры класса `DatagramSocket`.

Посылка *дейтаграмм* осуществляется методом `send()`, прием — методом `receive()`. Параметром этих методов служит дейтаграмма — экземпляр класса `DatagramPacket`. После получения дейтаграммы, из нее нужно извлечь сообщение методом `getData()`.

В листинге 1.5 дана схема клиента, посылающего дейтаграмму.

Листинг 1.5. Схема UDP-клиента

```
import java.net.*;
import java.io.*;

public class Sender{
    public static void main(String[] args){
        try{
            String message = "Привет!";

            // Преобразуем строку Unicode в массив байтов Cp866.
            byte[] data = message.getBytes("Cp866");

            InetAddress addr = InetAddress.getByName("www.bhv.ru");

            DatagramPacket pack =
                new DatagramPacket(data, data.length, addr, 1234);

            DatagramSocket ds = new DatagramSocket();

            ds.send(pack);

            ds.close();

        }catch(Exception e){
            System.out.println("From client: " + e);
        }
    }
}
```

В листинге 1.6 показана схема сервера, который ждет дейтаграмму, прослушивая порт 1234. После получения дейтаграммы сервер извлекает из нее сообщение и отправляет его в свой стандартный вывод.

Листинг 1.6. Схема UDP-сервера

```
import java.net.*;
import java.io.*;

class Recipient{
    public static void main(String[] args){
        try{
            DatagramSocket ds = new DatagramSocket(1234);

            while(true){
                DatagramPacket pack =
                    new DatagramPacket(new byte[2048], 2048);

                ds.receive(pack);

                System.out.println(new String(pack.getData(), "Cp866"));

            }
        }catch(Exception e){
            System.out.println("From server: " + e);
        }
    }
}
```

Новые средства Java API

В следующем выпуске JDK, его версия 1.1, число сетевых средств сильно возросло. Была разработана спецификация компонентов JavaBeans, которые могут совершенно самостоятельно "путешествовать" по сети и встраиваться в приложения, работающие на удаленной машине, как "родные" компоненты. Эта совместимость и взаимозаменяемость достигается четко описанным стандартным интерфейсом, который должен предоставлять каждый JavaBean. Описание содержится в спецификации JavaBeans API Specification, которая доступна по адресу <http://java.sun.com/products/javabeans/docs/spec.html>. В настоящее время все графические компоненты библиотек AWT и Swing, многие классы и целые приложения оформляются как JavaBeans. Классы, помогающие создавать JavaBeans, составили пакет java.beans.

Для того чтобы передавать уже существующие объекты Java по сети, было введено понятие *сериализации* — записи объекта в виде последовательности