Вадим Будилов

bhv

Интернет-программирование на Java

- Разработка Web-приложений
- Создание клиентских и серверных приложений
- Разработка сервлетов и JSP-страниц
- Серверные компоненты EJB и среда J2EE

МАСТЕР_ПРОГРАММ_

Вадим Будилов

Интернет-программирование на Java

Санкт-Петербург

«БХВ-Петербург»

2003

УДК 681.3.06
ББК 32.973.26-018.1
Б90

Будилов В. А.

Б90 Интернет-программирование на Java. — СПб.: БХВ-Петербург, 2003. — 704 с.: ил.

ISBN 5-94157-272-7

В книге подробно рассматриваются методы создания интернет-приложений на языке Java, в том числе Web-приложений, апплетов, серверных приложений, использование серверных страниц Java, конструирование и программирование пользовательских библиотек ярлыков Java, а также разработка приложений с применением современных технологий, реализованных в пакете J2EE. Подробно описано функционирование сервера Blazix. Внимание акцентировано на раскрытие наиболее существенных сторон создания клиентских и серверных приложений. Многочисленные примеры делают изложенный материал весьма наглядным и помогают его лучшему усвоению. Книга рассчитана на читателя, знакомого с программированием и имеющего некоторый опыт создания программ на любом языке.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Консультант	*Ильдар Хабибуллин*
Редактор	*Ирина Радченко*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Оформление серии	*Via Design*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 23.05.03.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 56,76.
Тираж 3000 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 5-94157-272-7

Содержание

Виктории Будкевич

Введение

Данная книга посвящена программированию на языке Java. Этот язык изначально создавался для программирования в сети Интернет, поэтому он является наиболее приспособленным к нуждам сети механизмом, использование прогрессивных методов которого позволяет реализовать наиболее эффективные, надежные, многократно используемые приложения, создаваемые на основе компонентов. Современные технологии программирования распределенных приложений основаны на детально разработанных концепциях, воплощенных средствами Java. Разработка компонентов не будет составлять большого труда, если задача сформулирована четко, а менеджер и разработчик знакомы со средствами, которые предоставляет язык Java.

В книге рассматриваются ставшие классическими подходы, используемые для создания серверных приложений — сервлетов и серверных страниц JSP (Java Server Pages). Для того чтобы сделать материал наиболее наглядным и легко воспринимаемым, читателю предлагается разбор примеров. Сама же книга посвящена изучению общих принципов рассматриваемых технологий без акцента на конкретной реализации той или иной технологии. Для облегчения изложения примеры работают с сервером Blazix, выбор этого сервера обусловлен тем, что он достаточно прост в использовании, поэтому не потребуется большого времени для изучения его устройства. Сервер Blazix представляет собой сервер, который исполняет функции Web-сервера и который работает с компонентами EJB (компонентами Enterprise JavaBeans), сервлетами и JSP. Это полнофункциональный сервер, поддерживающий возможности Java Application Server.

Книга рассчитана на читателя, знакомого с программированием и имеющего некоторый опыт создания программ на любом языке программирования. Книга может быть полезна студентам, а также тем, кто желает расширить свой кругозор и познакомиться с методами создания интернет-приложений на языке Java. Предполагается, что читатель знаком с основами программирования для сети Интернет. В современных интернет-технологиях широко применяются языки XML (eXtensible Markup Language — расширяемый язык разметки) — язык разметки, позволяющий дополнительно определять теги

(разметку), а также связанные с ними языки описания типов документов и языки стилей, например, XML Schema и XSL (eXtensible Stilesheets Language — расширяемый язык стилей). Все, кто связан с интернет-программированием, в той или иной степени знакомы с этими языками. Базовых знаний о них достаточно для работы с предлагаемой книгой. Для более детального знакомства с перечисленными языками советуем обратиться к специальной литературе. Java-сервера используют платформонезависимые форматы описания конфигурации как самого сервера, так и размещаемых на нем приложений. Основным используемым форматом является формат XML. Описатель размещения Web-приложений и компонентов EJB представляет собой XML-документ.

Тот факт, что конфигурация серверов Java-приложений представлена в формате XML, накладывает ряд требований. Современный язык Java имеет набор средств для работы с XML, а также с протоколами и технологиями, основанными на XML, например, с SOAP (Simple Objects Access Protocol — простой протокол доступа к объектам). Большое значение во всех серверах уделяется описанию служб Web, доступ к которым может быть осуществлен различными способами. Один из универсальных подходов — это использование протокола SOAP, который является легковесным протоколом, а сообщения на основе SOAP могут быть переданы поверх различных протоколов более низкого уровня, в том числе с использованием протокола HTTP (HyperText Transfer Protocol) или SMTP (Simple Mail Transfer Protocol — простой протокол передачи почты). Протокол SOAP естественным образом предполагает наличие SOAP-сервиса, расположенного на соответствующем сервере. Ввиду того, что протокол SOAP прост, его реализация собственными силами не представляет сложную задачу. Существуют и готовые реализации SOAP. Доступ к службам SOAP осуществляется по протоколу SOAP. В Интернете могут существовать различные службы, в том числе службы, доступ к которым основан на SOAP. Существует необходимость задания описания различных служб. В настоящее время консорциум W3C (World Wide Web Consortium) ведет работу по разработке серии языков нового поколения, которые сделают работу в сети Интернет более структурированной функционально. Выполнение частей приложений на различных компьютерах сети станет более естественным. Для этого, в частности, создается язык описания служб сети WSDL (Web Services Description Language — язык описания Web-служб).

Новые возможности сети Интернет становятся реальностью только тогда, когда есть механизмы их реализации. Высокоуровневое взаимодействие ресурсов сети Интернет друг с другом, которое не зависит ни от типа программного обеспечения, ни от архитектуры аппаратного обеспечения, возможно только тогда, когда существуют эффективные средства решения стоящих перед разработчиком задач на более низком уровне. Именно изучению таких методов посвящена предоставляемая вашему вниманию книга.

Она состоит из пяти глав и пяти приложений. *Первая глава* посвящена работе с апплетами — небольшими программами, которые работают на клиентском браузере в составе HTML-страницы (страницы, написанной на HyperText Markup Language — языке разметки гипертекста). Во *второй главе* рассмотрены вопросы сериализации и работы с сетью. *Третья глава* посвящена изучению технологии серверных страниц JSP (с использованием описаний на основе XML), рассмотрены способы создания пользовательских ярлыков JSP и создания классов обработчиков этих ярлыков. В этой же главе дано описание сервера Blazix. В *четвертой главе* более подробно рассмотрена технология создания сервлетов, а также устройство сервлета, его жизненный цикл, методы вызова контейнером сервлета в течение его жизненного цикла. *Пятая глава* посвящена изучению серверных компонентов EJB. После изучения этой главы читатель сможет ориентироваться в технологии создания компонентов EJB, разрабатывать и использовать компоненты EJB в собственных приложениях.

Глава 1

Интернет и Java

So, naturalists observe, a flea
Hath smaller fleas that on him prey;
And these have smaller still to bite'em;
And so proceed ad infinitum.

Sir Andrew Larsh

Java — это алгоритмический язык, который является разработкой компании Sun Microsystems и предназначен для программирования в сети Интернет, или, другими словами, язык, предназначенный для создания приложений, работающих, в частности, в сети WWW (World Wide Web). Наиболее компактными Java-приложениями являются *апплеты*. Апплеты — это небольшие программы, которые, как правило, создаются для работы в браузере. Апплеты могут существенно обогатить внешний вид HTML-страницы. При работе с ними удобно использовать пакеты, предоставляющие возможность работы с графикой и звуком. Значительное место в апплетах может занимать код обработки событий.

Для того чтобы научиться создавать апплеты, необходимо иметь минимальный навык программирования на языке Java.

1.1. Первое приложение на языке Java

Для того чтобы написать, скомпилировать и запустить приложение на языке Java, необходимо кроме файла с программой иметь среду выполнения программ Java. Для платформы, с который вы работаете, например, для Windows, следует получить пакет разработчика jdk1.3 или jdk1.4, который можно найти на сайте производителя **http://java.sun.com**. Пакет следует установить на компьютере в соответствии со всеми инструкциями, в частности необходимо указать значение переменной `path`, соответствующее папке, где установлен пакет Java. После того как пакет установлен, можно приступить к созданию простого кода (листинг 1.1).

Листинг 1.1. Файл VsemPrivet.java

```
class VsemPrivet {
 public static void main(String[] args) {
   System.out.println("Vsem Privet I S Dobrym Utrom!");
 }
}
```

Что здесь важно? Файл содержит класс, имя которого должно совпадать с именем файла. Класс этот содержит метод `main()`. Метод `main()` должен присутствовать всегда.

После того как файл с классом создан, вызовем компилятор javac из командной строки (рис. 1.1):

```
E:\temp>javac vsemprivet.java
```

Рис. 1.1. Вызов компилятора из командной строки

Вызов компилятора необходимо осуществить из той папки, где расположен файл VsemPrivet.java. Если компиляция прошла успешно, то сообщений об ошибках не будет. В папке появится новый файл с именем VsemPrivet.class. Этот файл содержит скомпилированный Java-код. Для выполнения классов Java-кода необходима среда выполнения Java или виртуальная машина Java. А для того чтобы среда выполнения классов Java была правильно инициализирована, необходимо указать путь к файлам классов в переменной `classpath`. Из командной строки это делается так:

```
set classpath=%classpath%;путь_к_папке_файла_класса
```

Теперь можно запустить программу (рис. 1.2), выполнив команду

```
java VsemPrivet
```

при этом расширение файла в имени класса не указывается.

Рис. 1.2. Запуск программы

Фактически при вызове среды выполнения Java в качестве аргумента указывается класс, а не файл. Класс хранится в файле, имя которого соответствует имени класса. Следует помнить, что Java различает строчные и прописные буквы. Если изменить регистр букв при обращении к классу, то Java выведет сообщения об ошибках (рис. 1.3).

Рис. 1.3. Появление ошибок вызвано неверным написанием имени класса

1.2. Апплеты

Апплет можно определить как небольшую программу, написанную на языке Java и работающую в браузере. В действительности апплет — это неполная, несамостоятельная программа, к тому же она вовсе не обязана быть маленькой. Небольшой ее создают только в целях удобства, по причине того, что класс программы, как правило, передается по сети, да и ресурсы браузера бывают ограничены. Кроме того, существуют способы использовать апплет не только в браузере. Просто для апплета типично быть маленькой программой,

выполняемой в браузере. Итак, наиболее корректным будет следующее определение: апплет — это программа, созданная на основе класса `java.applet.Applet` или на основе одного из подклассов этого класса.

Апплет — наследник графического интерфейса пользователя, это графический компонент, который может быть показан в окнах, например, в окне браузера или в другой программе. При этом апплет занимает некоторую часть окна браузера, и внутри апплета могут быть расположены другие графические элементы (кнопки, текст, поля и т. п.), отображены рисунки, линии, апплет может реагировать на события (например, на щелчки мыши), включать в себя другие элементы.

Сам по себе класс `Applet` не будет для нас слишком интересен. В окне он может создать пустую прямоугольную область, в которой ничего нет, область, которая не реагирует ни на какие действия. При создании апплета мы задаем класс на основе класса `Applet`, который содержит в себе дополнительные возможности, "оживляющие" апплет. В классе `Applet` определено несколько методов, которые необходимо переопределить во время создания апплета, изначально эти методы ничего не выполняют.

Отметим, что в апплете нет функции `main()`, это происходит потому, что апплет не является самостоятельной программой, он должен работать в составе другого приложения. Задачей разработчика является создание "ответов" на "запросы" "системы", то есть ответов на запросы той программы, с которой работает апплет.

Рассмотрим некоторые важнее моменты, часто используемые при создании апплетов. Один из описанных в классе `Applet` методов, которые, однако, изначально не приспособлены к каким-либо действиям, — метод `paint()`. Этот метод вызывается, когда необходимо что-либо нарисовать. В апплете метод `paint()` можно вызывать при необходимости вывода графических элементов, например, прямоугольников, линий, текста. Метод `paint()` определяется следующим образом:

```
public void paint(Graphics g) {
  // функции рисования
}
```

В качестве аргумента функции указывается параметр `g`, который имеет тип `Graphics`. Большинство задач по отображению в языке Java решаются с использованием объектов `Graphics`. Для работы с этими объектами существует множество методов.

1.2.1. Первый апплет

Создадим простой апплет, пусть этот апплет выводит текст "`Privet Vsem, a ne Hello World!`". Эта строка будет выводиться с использованием метода `paint()`. Чтобы не использовать полные имена методов, включая имена

пакетов (`Applet` вместо `java.applet.Applet` и `Graphics` вместо `java.awt.Graphics`), импортируем в начале файла основные пакеты (листинг 1.2).

Листинг 1.2. Файл PrivetVsemApplet.java

```
import java.awt.*;
import java.applet.*;
public class PrivetVsemApplet extends Applet {
  // Апплет для вывода строки текста
  public void paint(Graphics g) {
    g.drawString("Privet Vsem, a ne Hello World!", 15, 45);
  }
}
```

Сейчас необходимо скомпилировать класс. В действительности, рисование как таковое осуществляет метод `drawString()`, определенный в классе `Graphics`. В качестве параметров этого метода указывается строка, которая будет выведена, а также точка (координаты) апплета, где эта строка будет расположена. Теперь можно использовать апплет, создав новый объект, например, с помощью инструкции

```
Applet na = new PrivetVsemApplet();
```

Так можно поступать, если существует необходимость вставить апплет в окно другой программы. Но чаще всего апплеты используются в браузере. Для этого необходимо создать HTML-страницу, например, так, как это сделано в файле `PrivetVsemApplet.html` (листинг 1.3).

Листинг 1.3. Файл PrivetVsem.html

```
<applet code="PrivetVsemApplet.class" width=300 height=150>
</applet>
```

Здесь мы пренебрегли (для сохранения предельной простоты) правилами хорошего тона, создав простейший файл HTML, в котором не указаны основные его элементы. Это не страшно. Браузер автоматически восполнит недостающее. Поместим этот файл в ту же папку, где расположен класс апплета. Загрузим HTML-страницу в браузер. Если в браузере включена поддержка Java, то в окне получим отображенный в апплете текст (рис. 1.4).

Этот текст отображается в прямоугольнике размером 300×150 пикселов. Этот прямоугольник не видно, так как его цвет совпадает с цветом фона браузера.

Рис. 1.4. Отображение апплетом текста в окне браузера

В классе `Applet` существует метод `init()`, который выполняется сразу после того, как апплет будет создан, но до того, как будет отображен в окне. В этот момент можно инициализировать апплет. Разработчику не надо беспокоиться о вызове этого метода, он вызывается автоматически, необходимо лишь описать этот метод, создать нужные функции для него (листинг 1.4).

Листинг 1.4. Метод `init()`

```
public void init() {
  // Функции инициализации апплета
}
```

Инициализация апплета может быть осуществлена также с использованием функции конструктора. Однако при вызове конструктора недоступным будет размер апплета, в то время как функция `init()` позволяет использовать параметры размера апплета. Как правило, инициализация апплета осуществляется с применением метода `init()`.

В качестве примера использования метода `init()` перепишем наш пример и зададим цвет фона (листинг 1.5).

Листинг 1.5. Файл VsemPrivetApplet2.java

```
import java.awt.*;
import java.applet.*;
  public class VsemPrivetApplett2 extends Applet {
  public void init() {
    // инициализируем апплет, цвет фона — зеленый
    // цвет основной — красный.
    setBackground(Color.green);
    setForeground(Color.red);
   }
  public void paint(Graphics g) {
    g.drawString("Vsem Privet, a ne Hello World!", 30, 90);
  }
 }
```

Рис. 1.5. Инициализация апплета в методе `init()`

Сейчас прямоугольник апплета стал видимым и зеленым, а текст выведен красным цветом (рис. 1.5).

1.2.2. Апплеты AWT и апплеты Java 2

В примерах, рассмотренных выше, мы использовали пакет `java.awt.*`. Недостатки, обнаруженные в этом пакете, привели к созданию пакета `Swing`, явившегося составной частью версии Java 2 начиная с jdk1.2. Существует возможность создания апплетов на основе пакета `Swing`. Однако при этом нужно иметь в виду, что некоторые версии браузеров не будут поддерживать такие апплеты, к тому же апплеты, основанные на пакете `Swing`, отнимают большие ресурсы, чем апплеты, основанные на AWT (Abstract Window Toolkit, абстрактный оконный интерфейс).

Пакет `javax.swing` содержит класс `JApplet`. Класс `javax.swing.JApplet` можно использовать в качестве базового класса при создании апплетов. Класс `JApplet` является подклассом по отношению к классу `Applet`, т. е. апплеты `Japplets` являются полноценными апплетами, но возможности использовать `Swing` у них встроены. Рисование при помощи апплетов `JApplet` несколько более усложнено. Для класса, основанного на `JApplet`, не нужно писать метод `paint()`, зато здесь необходим метод инициализации апплета `init()`. Рисование же осуществляется при помощи вставки компонентов.

Первый апплет JApplet

Рассмотрим пример апплета, в котором проявляются основные идеи использования пакета `Swing` и программирования графического интерфейса. Программирование элементов графики предполагает использование готовых компонентов, например, таких как кнопки и т. п., посредством которых апплет может взаимодействовать с пользователем. В примере (листинг 1.6) апплет состоит только из кнопки (листинг 1.7).

Листинг 1.6. Код апплета находится в файле Knopka.java

```
import javax.swing.*;  // классы Swing GUI
import java.awt.event.*; // классы обработки событий
  public class Knopka extends JApplet implements ActionListener {
  public void init() {
    // метод вызывается после создания апплета,
    // но перед тем, как апплет будет отображен в окне
    JButton knopka = new JButton("Nazhmi Menya!");
    knopka.addActionListener(this);
    getContentPane().add(knopka);
```

```
  }
  public void actionPerformed(ActionEvent evt) {
  // Метод используется при наступлении события.
  // В данном случае возможно только одно событие –
  // нажатие кнопки.
  // В ответ отображается диалоговое окно
  // с информацией и кнопкой OK,
  // нажатие которой закрывает новое диалоговое окно.
  String title = "Priovetstvuyu vas";  // заголовок диалогового окна
  String message = "Privet vam ot biblioteki Swing.";
  JOptionPane.showMessageDialog(null, message, title,
  JOptionPane.INFORMATION_MESSAGE);
  }
}
```

Этот апплет вызывается из файла Knopka.html.

Листинг 1.7. Файл Knopka.html

```
<p align=center>
  <applet code="Knopka.java" width=250 height=140 >
  </applet>
</p>
```

При загрузке этого апплета в окно браузера появляется кнопка размером 250 × 140 пикселов, нажатие которой создает окно с информацией и кнопкой **OK** (рис. 1.6).

Кнопка создается на основе класса кнопок `JButton` (класс `javax.swing.JButton`). После создания апплета кнопка также должна быть создана и вставлена в апплет. Эта задача входит в перечень задач, выполняемых при инициализации апплета. Кнопка создается при помощи следующей функции:

```
JButton knopka = new JButton("Nazhmi Menya!");
```

В качестве параметра конструктору передается строка, которая будет отображена на кнопке. Кнопка должна быть вставлена в контекст панели апплета с помощью метода

```
getContentPane().add(knopka);
```

Нажатие кнопки создает событие, тип которого `ActionEvent`. Для обработки событий этого типа создаем метод

```
public void actionPerformed(ActionEvent evt) { ... }
```

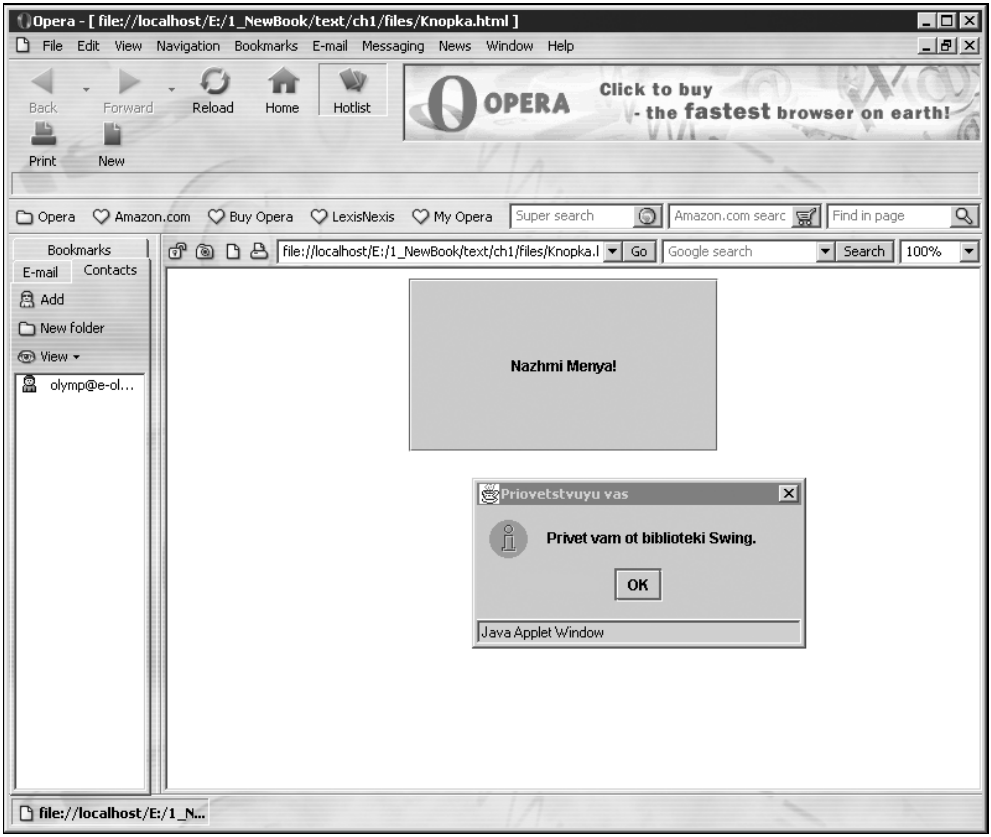

Рис. 1.6. Функционирование апплета Swing

Кнопка создается на основе класса кнопок `JButton` (класс `javax.swing.JButton`). После создания апплета кнопка также должна быть создана и вставлена в апплет. Эта задача входит в перечень задач, выполняемых при инициализации апплета. Кнопка создается при помощи следующей функции:

```
JButton knopka = new JButton("Nazhmi Menya!");
```

В качестве параметра конструктору передается строка, которая будет отображена на кнопке. Кнопка должна быть вставлена в контекст панели апплета с помощью метода

```
getContentPane().add(knopka);
```

Нажатие кнопки создает событие, тип которого `ActionEvent`. Для обработки событий этого типа создаем метод

```
public void actionPerformed(ActionEvent evt) { ... }
```

Кнопке необходимо сообщить, что апплет прослушивает связанные с ней события. Это делается при помощи метода `addActionListener()`, который используется внутри метода `init()`.

Что происходит внутри метода `actionPerformed()`? При нажатии кнопки должно появиться окно с информацией. Это легко осуществимо средствами пакета `Swing`. Класс `swing.javax.JOptionPane` содержит статический метод `showMessageDialog()`, с помощью него и решается поставленная задача.

В этом примере апплет сам заботится о том, чтобы прослушивать сообщения кнопки. Однако это не самый лучший способ. Для прослушивания можно выделить специальный объект, который будет отвечать за обработку событий. Это осуществляется следующим образом (листинг 1.8).

Листинг 1.8. Файл knopka2.java

```
import javax.swing.*;
import java.awt.event.*;
  public class Knopka2 extends JApplet {
    public void init() {
      JButton knopka = new JButton("Nazhmi Menya!");
      knopka.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
          // метод вызывается при нажатии кнопки
          String title = "Privet";  // название кнопки
          String message = "Esche raz privet ot Swing.";
          JOptionPane.showMessageDialog(null, message, title,
          JOptionPane.INFORMATION_MESSAGE);
        }
      }
    );
    getContentPane().add(knopka);
  }
```

1.3. Кратко о HTML

Наиболее распространенное применение апплетов — использование их в HTML-страницах. *HTML-страница* — это файл, содержащий в себе код, созданный с использованием языка HTML, HyperText Markup Language (язык разметки гипертекста). При помощи этого языка описывается содержимое Web-страниц, которое загружается в клиентскую программу-браузер. Сам по себе HTML-код выглядит совсем не так, как будет представлена

страница с использованием HTML-кода, загруженного в браузер. Помимо самого текста, HTML-код будет содержать инструкции, которые определят структуру текста, внешний вид страницы, динамические элементы, вставленные в страницу, рисунки и т. п. При помощи HTML в страницу могут быть вставлены и Java-апплеты.

Создать HTML-страницу можно при помощи простого текстового редактора, вводя вручную все инструции разметки. Однако существует большое количество программ, которые помогают создавать HTML-страницы без необходимости вникать в детали HTML-кода. При этом создание HTML-страницы становится столь же простым делом, как создание текстовой страницы в текстовом процессоре, например, в программе MS Word. Однако использование программ автоматического редактирования HTML-страниц не позволяет воспользоваться всеми возможностями, предоставляемыми языком HTML, к тому же динамические элементы требуют задания определенных параметров, которые необходимо указывать в контексте потока HTML-кода. Редакторы HTML-страниц порой генерируют чрезмерно причудливый HTML-код, который требует редактирования вручную. Все это говорит о том, что знание языка HTML просто необходимо.

В этом разделе мы рассмотрим основы языка HTML.

Команды языка HTML представляют собой теги. Структура тега такова:

```
<имя_тега [модификаторы-атрибуты]>
```

Имя тега — это слово. Существует ограниченный набор стандартных имен тегов. Модификаторы-атрибуты — это определенные свойства, в каждом теге их может быть указано несколько или же может не быть ни одного. Для каждого тега существует свой набор атрибутов. Модификаторы имеют такой вид:

```
Имя_атрибута = значение
```

Как правило, значение заключается в кавычки. Это особенно важно в том случае, если строка значения состоит из нескольких слов. Помните, что HTML не чувствителен к регистру, то есть заглавные и строчные буквы несут один и тот же смысл. Вот пример тега: `<HR>` — он используется для создания горизонтальной черты-линейки, проходящей слева направо (или, если хотите, справа налево) через всю страничку. Тег `<HR>` может иметь несколько атрибутов, например, атрибуты `WIDTH` и `ALIGN`. Для того чтобы создать короткую линейку, расположенную по центру странички, используем следующий тег:

```
<HR align=center width="33%">
```

Здесь указана ширина линейки в процентах от ширины всей странички (рис. 1.7). Вместо относительной ширины можно задать значение ширины линейки в пикселах. Атрибут `ALIGN` может принимать значения `CENTER` (или `CENTRE`), `LEFT`, `RIGHT`.

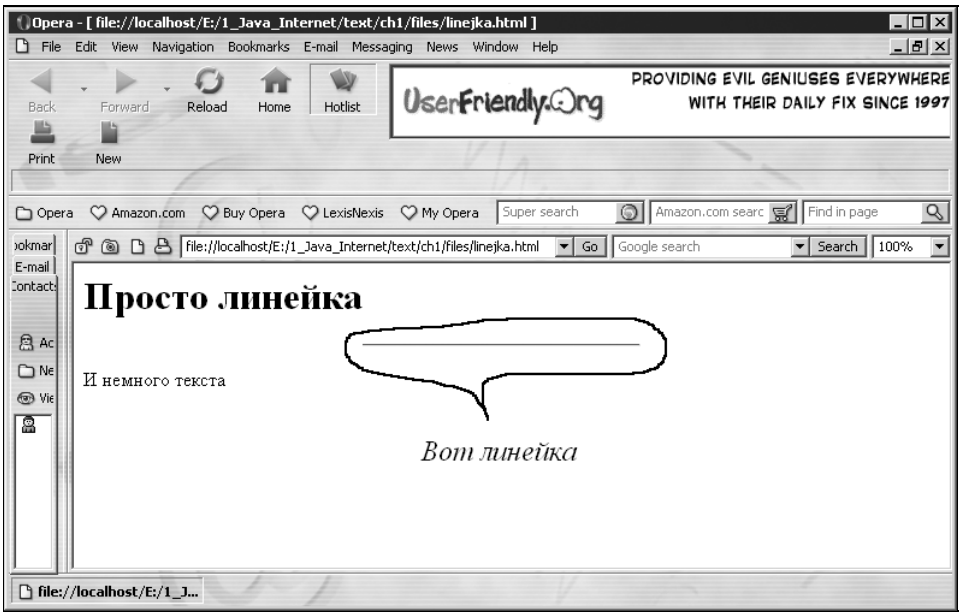

Рис. 1.7. Простая линейка и текст HTML

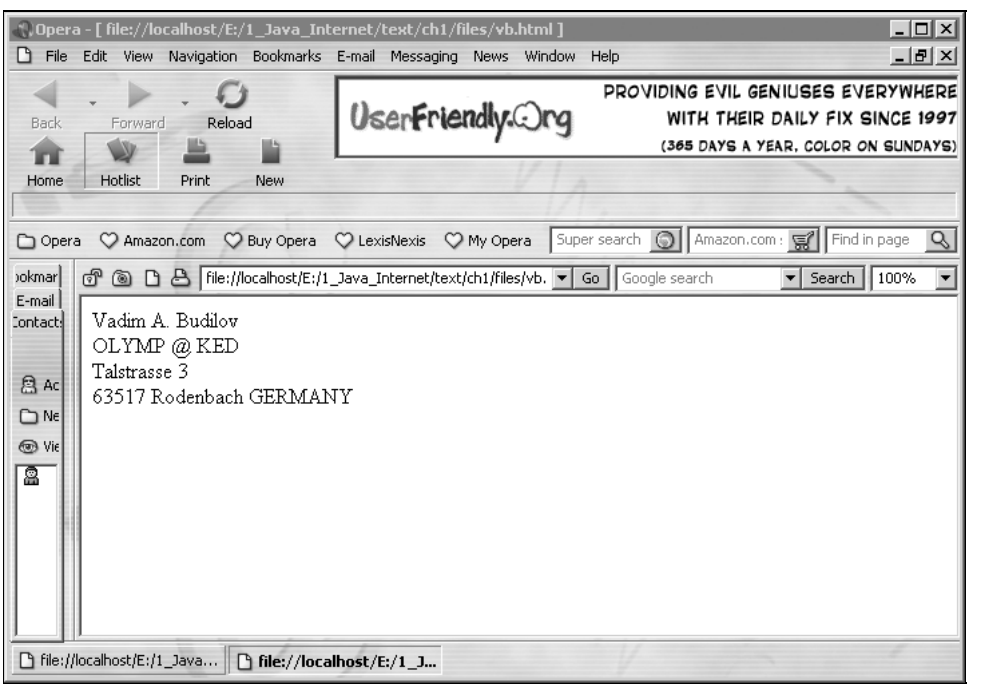

Рис. 1.8. Перенос строки

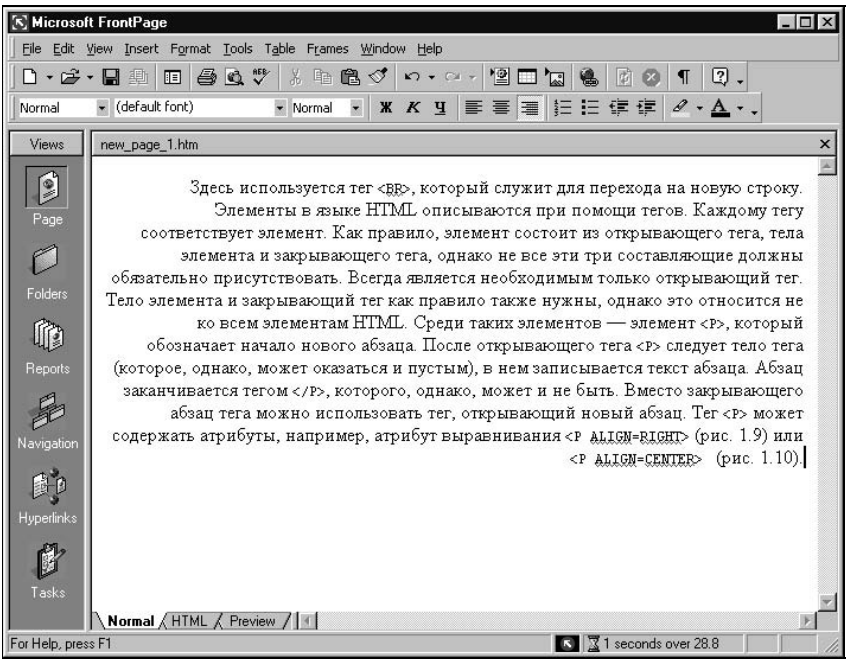

Рис. 1.9. Выравнивание по правой стороне

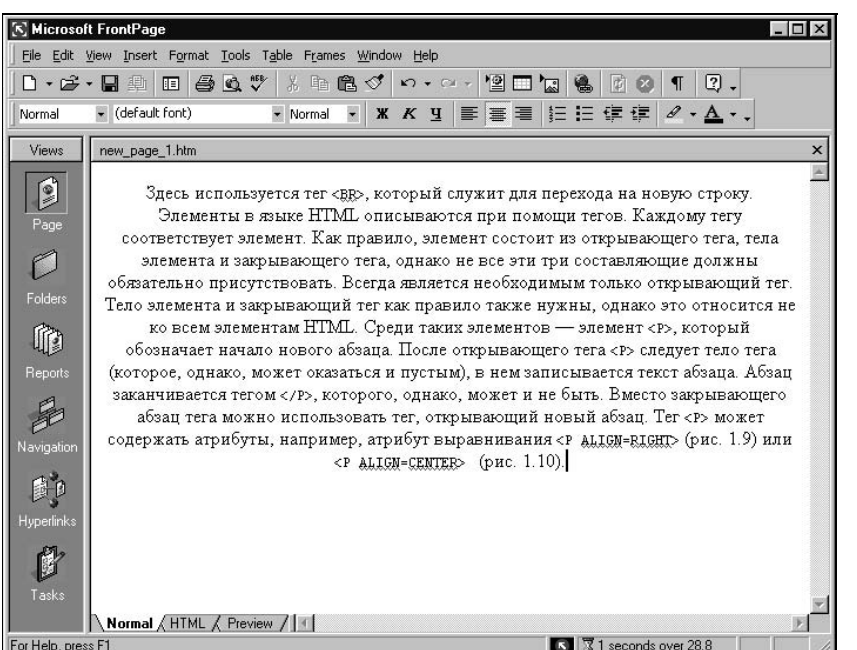

Рис. 1.10. Выравнивание по центру

Большинство тегов требует наличия парных им закрывающих тегов, которые выглядят так:

```
</имя_тега>
```

Однако не все теги таковы, некоторые элементы не требуют использования закрывающих тегов, например (рис. 1.8):

```
Vadim A. Budilov<BR>
OLYMP @ KED<BR>
Talstrasse 3<BR>
63517 Rodenbach
GERMANY<BR>
```

Здесь используется тег
, который служит для перехода на новую строку. Элементы в языке HTML описываются при помощи тегов. Каждому тегу соответствует элемент. Как правило, элемент состоит из открывающего тега, тела элемента и закрывающего тега, однако не все эти три составляющие должны обязательно присутствовать. Всегда является необходимым только открывающий тег. Тело элемента и закрывающий тег как правило также нужны, однако это относится не ко всем элементам HTML. Среди таких элементов — элемент <P>, который обозначает начало нового абзаца. После открывающего тега <P> следует тело тега (которое, однако, может оказаться и пустым), в нем записывается текст абзаца. Абзац заканчивается тегом </P>, которого, однако, может и не быть. Вместо закрывающего абзац тега можно использовать тег, открывающий новый абзац. Тег <P> может содержать атрибуты, например, атрибут выравнивания <P ALIGN=RIGHT> (рис. 1.9) или <P ALIGN=CENTER> (рис. 1.10).

Далее будут рассмотрены очень кратко основные элементы HTML.

1.3.1. Общая структура HTML-документа

HTML-документ имеет стандартную структуру. Документ состоит из элемента <HTML>, он открывается тегом <HTML> и заканчивается тегом </HTML>. Между этими тегами располагается заголовок документа, обозначаемый при помощи тегов <HEAD> и </HEAD>. Основное содержание HTML-документа располагается между тегами <BODY> и </BODY> (листинг 1.9).

Листинг 1.9. Общая структура HTML-документа

```
<HTML>
  <HEAD>
    <TITLE>
      Заголовок страницы
    </TITLE>
```

```
  </HEAD>
  <BODY>
    Содержимое страницы
  </BODY>
</HTML>
```

Тег <BODY> имеет множество атрибутов, например,

```
<BODY bgcolor=white>
```

Этот тег указывает, что цвет фона страницы будет белый. Помимо этого существуют и другие атрибуты, например, цвет текста (TEXT), цвет ссылок (LINK), цвет посещенных ссылок (VLINK):

```
<BODY bgcolor=black text=white link=blue alink=red vlink=gray>
```

Какие атрибуты связаны с тем или иным HTML-элементом? Ответ на этот вопрос можно найти в документации по HTML или в справках в специализированных редакторах, например, в редакторе SlickEdit.

Вот список всех атрибутов элемента <BODY>, определенных в HTML 4. Однако это не значит, что те или иные браузеры не могут определить и дополнительные атрибуты для элемента <BODY>, значительно расширив перечень используемых атрибутов. Еще раз повторим, что это лишь те атрибуты элемента <BODY>, которые описаны в стандарте HTML 4.

- background = url
- text = color
- link = color
- vlink = color
- alink = color
- id
- class
- lang
- dir
- title
- style
- bgcolor
- onload
- onunloadonclick
- ondblclick
- onmousedown
- onmouseup

- onmouseover
- onmousemove
- onmouseout
- onkeypress
- onkeydown
- onkeyup

Существует набор стандартных названий цветов.

- Black = #000000
- Green = #008000
- Silver = #C0C0C0
- Lime = #00FF00
- Gray = #808080
- Olive = #808000
- White = #FFFFFF
- Yellow = #FFFF00
- Maroon = #800000
- Navy = #000080
- Red = #FF0000
- Blue = #0000FF
- Purple = #800080
- Teal = #008080
- Fuchsia = #FF00FF
- Aqua = #00FFFF

Справа от названия указано соответствующее этому цвету значение в формате RRGGBB. Вместо названия цвета при задании цвета можно использовать этот формат.

1.3.2. Заголовки и шрифты

В HTML предусмотрено несколько элементов для оформления структуры текста. Для выделения заголовков используются теги <H1>, <H2>, ..., <H6>. Они соответствуют заголовкам соответствующих уровней. Эти теги всегда должны иметь парные им закрывающие теги, такие, как </H1>. Заголовки могут быть выровнены с использованием атрибута ALIGN со значениями LEFT, RIGHT, CENTER. Например:

```
<H1 align=center>Заголовок первого уровня</H1>
```