И. Шапошников

bhv
www.bhv.ru
www.bhv.kiev.ua

ИНТЕРНЕТ
ПРОГРАММИРОВАНИЕ
2-е издание

МАСТЕР
ПРАКТИЧЕСКОЕ РУКОВОДСТВО

Игорь Шапошников

ИНТЕРНЕТ
ПРОГРАММИРОВАНИЕ

2-е издание

Дюссельдорф ◆ Киев ◆ Москва ◆ Санкт-Петербург

УДК 681.3.06

Рассматриваются язык HTML и средства быстрой разработки Web-сайтов программы FrontPage 2000, а также методика создания CGI-приложений, ISAPI-расширений и элементов ActiveX с использованием Delphi 5. Отдельная глава посвящена технологии Java, созданию апплетов и внедрению их в HTML-документы при помощи JBuilder 2/3 и вопросам миграции на него с Delphi. Подробно описаны возможности расширяемого языка XML, управление процессом активизации ссылок, новые функции встроенных в XML технологий XLink и XPointer, языки сценариев VBScript и JavaScript.

Хотя книга и предназначена для разработчиков Web-сайтов и Интернет-приложений, знакомых с основами программирования и технологиями RAD, но благодаря доступной манере изложения материала и большому количеству примеров будет полезна также и для начинающих программистов.

Для разработчиков Web-сайтов и программистов

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Наталья Таркова*
Редактор	*Ольга Афанасьева*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Ангелины Лужиной*
Зав. производством	*Николай Тверских*

Шапошников И. В.

Интернет-программирование — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2000. — 368 с.: ил.

ISBN 5-94157-024-4

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 26.09.00.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 29,67.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

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

Содержание

Даниленко Ольге.
Все, что я делаю, я делаю для тебя.

Введение

"Компьютер без Интернета — деньги на ветер". Первый раз эту фразу я услышал в рекламе местного Интернет-провайдера. На мой взгляд, она абсолютно верна. Все больше и больше людей выходят в Интернет за информацией, все больше и больше коммерческих фирм предоставляют им эту информацию. Сегодня число пользователей Интернета составляет уже сотни миллионов. Вдумайтесь в эту цифру — *сотни миллионов*. В сложившейся ситуации все сильнее начинает ощущаться нехватка Web-программистов.

Обратите внимание, я сознательно не употребляю термина "Web-мастер". Хороший Web-мастер должен разбираться в дизайне, пользовательском интерфейсе, теории цвета и в прочих очень умных и нужных вещах. Я говорю об Интернет-программировании. Ведь для работающего "промышленного" Web-сайта необходимо постоянное обновление информации. Типичный пример — страница с перечислением товаров, находящихся на торговом складе фирмы с указанием цен, количества и других сопутствующих данных. А ведь наверняка у этой фирмы уже есть база данных, которая отслеживает текущую ситуацию. Задача "Web-программиста" — грамотно присоединить эту базу к Web-сайту.

Общее представление о задачах мы, кажется, получили. Теперь разберемся со структурой книги. В первой главе мы просто обязаны рассмотреть язык HTML. Согласитесь, изучать программирование для World Wide Web без его родного языка, по меньшей мере, странно. Впрочем, те, кто уже знаком с ним, могут безболезненно пропустить эту главу.

Во второй главе мы рассмотрим средство быстрого создания сайтов FrontPage 2000, разработанное фирмой Microsoft.

Третья глава будет посвящена обзору языка XML, который сейчас уже официально является преемником HTML. В этой же главе мы ознакомимся с XML-редактором.

В четвертой главе мы ознакомимся с возможностями языков встроенных сценариев JavaScript и VBScript.

Пятая глава будет предоставлена вопросам создания Web-приложений с помощью Delphi 5. Почему именно Delphi, а не, к примеру, Perl? Все очень просто. На сегодняшний день Delphi — одно из наиболее распространенных средств разработки (если не самое распространенное), и очень многим программистам будет намного проще узнать еще один способ работы в их родной среде, чем учить новый язык. К тому же, на данный момент приложения на Perl не отличаются высокой скоростью работы в силу присущих ему ограничений.

Ну, а в шестой части мы рассмотрим технологию Java. На сегодняшний день это, пожалуй, самая перспективная и многообещающая технология в мире Интернет-программирования. В качестве инструментального средства мы используем JBuilder3 все той же фирмы Borland.

Глава 1

Язык HTML

Начало

Как известно, Web-сайт хранит информацию в виде страниц, которые просматривают его посетители. Каждая такая страница (Web-page) должна быть создана в виде текстового файла на языке HTML (Hyper Text Markup Language). При соединении со страницей браузер получает файл и воспроизводит на его основе вид страницы. Эта технология позволяет отображать содержание Web-страниц почти одинаково в любом браузере.

Для создания Web-документов в настоящий момент существует множество программных продуктов, которые почти полностью избавляют пользователя от необходимости набирать текст HTML-файла вручную. Но для наших целей будет достаточно любого текстового редактора, такого как Блокнот или WordPad.

Итак, стандартная процедура создания Web-страницы заключается в следующем: набрать текст HTML-файла в любом оказавшемся под рукой текстовом редакторе, сохранить его с расширением html, открыть файл с помощью любого Интернет-браузера и насладиться собственным творчеством. Как видите, ничего сложного.

Разберемся с общей структурой языка HTML. Любой документ, написанный с помощью этого языка, состоит из содержимого страницы, то есть текста, и управляющих символов — тэгов. Все тэги HTML обязательно заключаются в угловые скобки (<>). Как правило, используется стартовый тэг и завершающий тэг. К примеру, первый и последний тэги любого HTML-документа обязательно выглядят как `<HTML>` и `</HTML>`. Как видно, завершающий тэг отличается от стартового наличием слеша.

Ну что ж, попробуем написать первую страничку. По сложившейся традиции, будем приветствовать окружающий мир.

Листинг 1.1. Файл First.html

```
<HTML>
 <HEAD>
  <TITLE>Моя первая страница!</TITLE>
```

```
 </HEAD>
 <BODY>
  Здравствуй, мир!! Ты меня слышишь?
 </BODY>
</HTML>
```

Результат открытия этого файла с помощью браузера Internet Explorer можно увидеть на рис. 1.1.

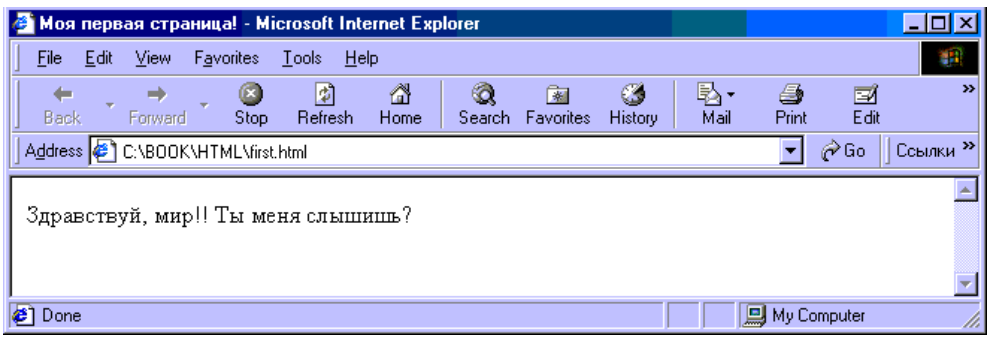

Рис. 1.1 Отображение браузером HTML-документа, приведенного в листинге 1.1

При анализе изображения страницы и текста HTML-документа можно заметить, что текст, между тэгами `<HEAD>` и `</HEAD>` помещается в заголовок окна браузера, а текст между тэгами `<BODY>` и `</BODY>` является содержимым страницы.

Конечно, кроме простых текстовых строк, нам хотелось бы иметь возможность писать тексты с заголовками, разбиением на абзацы, цитатами, управлять шрифтами, цветом и прочими разностями и "вкусностями".

Что касается заголовков, то с ними все просто. Заголовок выделяется тэгами `<Hx>` и `</Hx>`, где `x` — цифра, обозначающая уровень заголовка. Всего может быть 6 уровней.

Необходимо также иметь в виду, что браузер игнорирует символы возврата каретки и разбивает строки в зависимости от размера окна. Поэтому для создания отдельных абзацев используется тэг `<P>`. Строго говоря, для завершения абзаца необходимо использовать тэг `</P>`, однако он является необязательным, и в силу этого редко используется.

Мы привыкли в своих документах использовать различное выравнивание текста. Для этих целей, кстати, тоже используется тэг абзаца `<P>`. У этого тэга есть параметр `ALIGN`, который может принимать значения `LEFT`, `RIGHT` и `CENTER` для выравнивания текста абзаца по левому краю окна браузера, пра-

вому краю и, соответственно, центру окна. Для примера можно привести следующий текст:

```
<P ALIGN=RIGHT> Абзац, прижатый вправо
```

Однако здесь возникает одна маленькая проблема. Тэг <P ALIGN=CENTER> далеко не всегда работает так, как нам того хочется. Встречаются случаи, когда браузеры игнорируют его. Поэтому для центрирования текста используют тэги <CENTER> и </CENTER>.

Оформление текста

По вполне понятным причинам, в тексте документа HTML не указывается шрифт, которым будет отображаться текст документа. Ведь если вы привыкли использовать шрифт Crystal, совсем не обязательно пользователи Макинтошей будут иметь его у себя на машинах. Поэтому каждый браузер использует свои шрифты. Так что же, шрифтами управлять нельзя? Оказывается, можно. Просто наши возможности слегка ограничены.

Итак, у конечного пользователя текст отображается шрифтом, выбранным по умолчанию для его браузера. Прежде всего мы рассмотрим тэги, меняющие начертания шрифта. Для использования полужирного начертания шрифта предназначены тэги <B> и </B> (стиль Bold). Для получения курсива — <I> и </I>, подчеркнутый текст — <U> и </U>, перечеркнутый текст — <STRIKE> и </STRIKE>. Тэги <TT> и </TT> применяются для отображения текста моноширинным шрифтом. Примером такого шрифта может служить известный всем Courier.

Как и в случае с использованием конкретного шрифта, мы ограничены в прямом указании размера шрифта. Это связано с тем, что невозможно предсказать, в каком разрешении экрана будет работать пользователь, просматривающий нашу страницу. Если мы укажем слишком большой размер шрифта, информация может не уместиться на экране полностью, если укажем слишком маленький, на мониторах с высоким разрешением читать текст будет очень неудобно. Поэтому в HTML используются тэги, позволяющие лишь изменять базовый размер шрифта, выбранного в браузере по умолчанию.

Так, тэги <BIG> и <SMALL> в паре с закрывающими тэгами </BIG> и </SMALL> ответственны за увеличение и уменьшение размера текущего шрифта соответственно.

Для каждого документа существуют базовые значения параметров шрифта. Так, например, базовое значение размера шрифта можно задать тэгом <BASEFONT SIZE=n>, где n принимает значения от 1 до 7. По умолчанию это значение равно 3. Обратите внимание: это не привычный для нас всех размер шрифта в пунктах. Здесь размеру шрифта 1 соответствует размер шриф-

та 9 пунктов, а для размера 7 устанавливается шрифт размером 36 пунктов. Естественно, то, как будет выглядеть сам шрифт на отдельно взятом компьютере, предсказать невозможно, поэтому рекомендуется избегать крайних значений. Но это уже проблемы дизайна, а мы обещали себе не касаться их.

Изменять размер шрифта в теле документа можно с помощью команд `<FONT SIZE=+\-n>` и `</FONT>`. Все это можно проиллюстрировать следующим примером.

Листинг 1.2. Файл Second.html

```
<HTML>
 <HEAD>
  <TITLE>Моя вторая страница!</TITLE>
 </HEAD>
 <BODY>
  <P>
  <BASEFONT SIZE=4>
  Здесь мы напечатали обычный текст.
  <P>
  <FONT SIZE=+1>
  Здесь мы немного увеличили шрифт.
  </FONT>
  <P>
  Мы умеем печатать <BIG> большой </BIG> и <SMALL> маленький </SMALL>
текст.
  <P>
  Можем его <U>подчеркивать </U> и <STRIKE> перечеркивать</STRIKE>.
 </BODY>
</HTML>
```

Результат просмотра этого файла через браузер показан на рисунке 1.2.

Перечислим еще несколько возможностей отображения текста. Так, цитаты отображаются с помощью тэгов `<BLOCKQUOTE>` и `</BLOCKQUOTE>`. Для подстрочных и надстрочных индексов зарезервированы пары тэгов `<SUB>` — `</SUB>` и `<SUP>` — `</SUP>` соответственно.

Мы также имеем возможность вставлять в тело HTML-документа текст, написанный в каком-либо текстовом редакторе. Этот текст должен обрамляться тэгами `<PRE>` и его закрывающей парой `</PRE>`. Длина строки, установленной в этом текстовом редакторе, заранее неизвестна. Поэтому, чтобы не нарушать оформление текста, у тэга `<PRE>` был введен параметр `width`, который указывает длину строки. Таким образом, тэг `<PRE width=60>` позволя-

ет указывать, что для любого размера окна браузера при просмотре этого текста необходимо установить длину строки в 60 символов. По умолчанию, значение этого параметра равно 80.

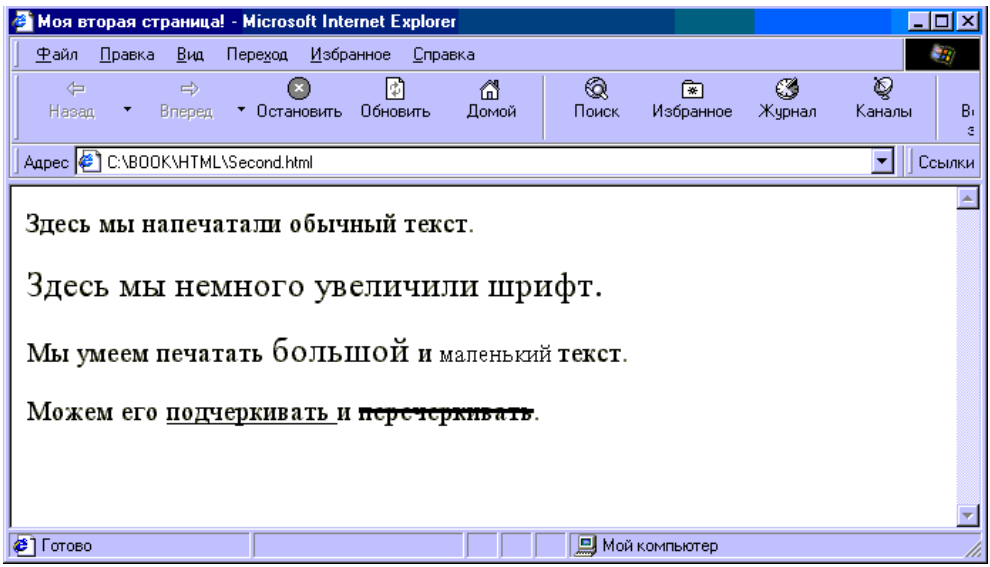

Рис. 1.2. Окно браузера с результатом просмотра файла, приведенного в листинге 1.2

Вполне вероятно, что стандартная цветовая гамма отображения HTML-текста (черный на сером) покажется кому-то несколько унылой. В таком случае используйте цветовое оформление текста и фона. Цвет текста задается тэгами `<FONT COLOR =#xxxxxx>` и `</FONT>`. Параметр `#xxxxxx` представляет собой шестизначное численное значение для цвета в формате RGB, где первые две цифры определяют насыщенность красного в требуемом цвете, вторые две — насыщенность зеленого, и последние две — насыщенность синего. Обратите внимание, цифры указываются в шестнадцатеричном формате.

Однако для отображения текста с применением стандартных шестнадцати цветов можно использовать и их словесное название. Например, тэг `<FONT COLOR=RED>` покажет текст красного цвета. Таблица соответствий названий цветов и их аналогов в русском языке приведена в Приложении 1. Для изменения цвета фона используется тэг `<BODY BGCOLOR=#xxxxxx>`.

В тексте документа нельзя напрямую использовать символы угловых скобок (<>), амперсант (&) и двойные кавычки ("), т. к. эти символы являются управляющими для языка HTML. Поэтому для их отображения используются последовательности символов `&lt`, `&gt`, `&amp` и `&quot`. Обратите внимание, в данном случае HTML чувствителен к регистру, и последовательности

надо набирать маленькими буквами. Конечно, на самом деле подобных символов достаточно много, и их список приведен в Приложении 2.

В самом конце этой главы посмотрим, как нарисовать горизонтальную линию. Действительно, в документах HTML очень часто блоки текста отделяют друг от друга тонкой горизонтальной линией. Для ее отображения используется тэг <HR COLOR=#xxxxxx SIZE=x WIDTH=y ALIGN=left|right|center NOSHADE>. Здесь любой параметр, кроме, естественно, HR, необязателен. Значение COLOR определяет цвет, SIZE указывает ширину разделительной линии в пикселах, ALIGN определяет выравнивание линии влево, вправо или по центру. По умолчанию линия центрируется. WIDTH указывает ширину линии в пикселах. В принципе, в параметре WIDTH можно указать размер линии в процентах от ширины окна, но тогда после цифрового значения необходимо поставить символ процента. Параметр NOSHADE заставляет рисовать линию без тени.

Списки, ссылки и графика

Возможностей для построения списков и перечислений HTML предоставляет более чем достаточно. Мы можем делать как нумерованные, так и ненумерованные списки, ставить номера в любом виде (римские и арабские цифры, варианты их написания), использовать различные виды маркеров ненумерованного списка, создавать списки с пояснениями. Для всего этого предусмотрены свои варианты действий.

Итак, для создания обычного ненумерованного списка (перечисление без упорядочивания) используется следующая конструкция: начало списка открывается тэгом <UL>, каждый элемент списка отмечается тэгом <LI>, а завершение списка делается с помощью закрывающего тэга </UL>. Для тэга <UL> есть параметр TYPE, который определяет вид маркеров списка. То есть, если мы зададим <UL TYPE=CIRCLE>, то в качестве маркера для пунктов списка будет использована окружность, значение параметра DISK задает в качестве маркера точку, а SQUARE — квадрат.

Если перед нами стоит задача воспроизвести нумерованный список, то придется воспользоваться тэгами <OL> и </OL>. У стартового тэга <OL> существуют параметры TYPE, задающий тип списка, и START, задающий начальный номер списка. Так, при использовании конструкции <OL TYPE=A> в качестве нумераторов списка будут использованы большие латинские буквы. Значение параметра TYPE=a заставит нумеровать список маленькими латинскими буквами, I — большие римские цифры (I, II,III,), i — маленькие римские цифры (i,ii,iii,), 1 — обычные арабские цифры.

Если потребуется создать список с пояснениями, то для этого используются тэги <DL> и </DL>, причем для обозначения самого элемента списка зарезервирован тэг <DT>, а для пояснения его — <DD>.

Впрочем, все это лучше один раз увидеть воочию, и сразу понять, как выглядят эти списки.

Листинг 1.3. Файл1-3.html

```
<HTML>
 <HEAD>
  <TITLE>Моя очередная страница!</TITLE>
 </HEAD>
 <BODY>
 Ненумерованный список
  <UL>
  <LI> Первый ненумерованный пункт
  <LI> Второй ненумерованный пункт
  </UL>
  Пронумерованный список
  <OL TYPE=I START=10>
  <LI> Первый пункт, помечен как десятый
  <LI> Второй пункт, помечен как одиннадцатый
  </OL>
  Список с пояснениями
  <DL>
  <DT> Нумерованный список
   <DD> Список, у которого все элементы пронумерованы
  <DT> Непронумерованный список
    <DD> Список, у которого все элементы просто помечены маркерами
  </DL>
 </BODY>
</HTML>
```

На рис. 1.3 видно, как будет выглядеть этот документ при просмотре его через браузер.

Теперь самое время перейти к ссылкам в документах HTML. При переводе полного названия этого языка самое первое слово — "гипертекстовый", то есть обеспечивающий ссылки на различные ресурсы. В HTML-документах используются локальные и внешние ссылки. Таким образом, мы можем сказать, что гиперссылка — это указание адреса HTML-документа, к которому можно перейти от текущего документа. Причем для пользователя во внутренних и внешних ссылках нет никакой разницы. Действия для перехода к

следующей главе читаемого документа и для перехода к страничке, находящейся где-нибудь на другом материке, абсолютно идентичны.

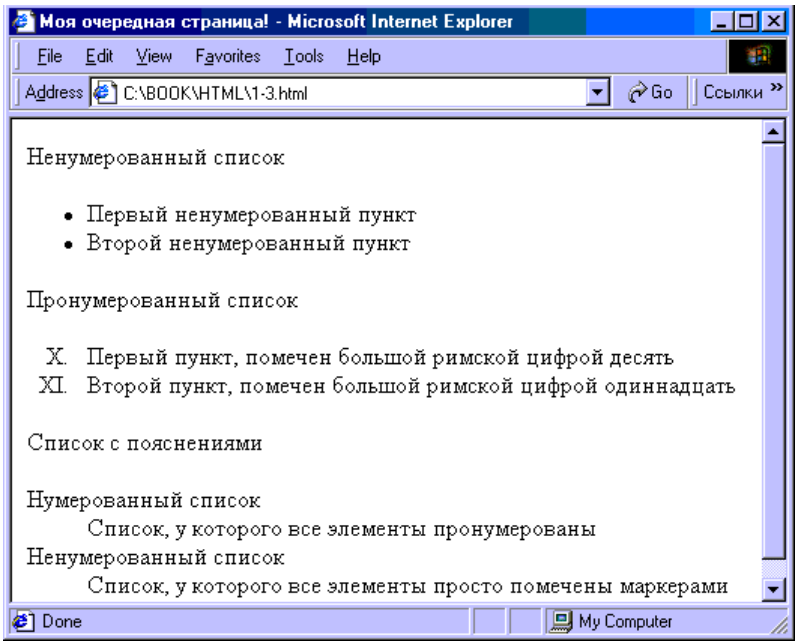

Рис. 1.3. Окно браузера с результатом просмотра файла, приведенного в листинге 1.3

Итак, для вставки ссылки в документ необходимо использовать тэги `<A HREF=URL>` и `</A>`, между которыми вписать текст самой ссылки. Он будет выделен цветом (обычно синим) и подчеркнут. Значение параметра `HREF` — это адрес той странички или того ресурса, на который мы, собственно, и ссылаемся. По поводу самого универсального идентификатора ресурсов — URL, и его структуры стоит поговорить особо. Сам URL записывается в виде `servis://server[:port][/path]`. В этом выражении `servis` — это наименование протокола, по которому мы обращаемся к ресурсу (например, `HTTP` — доступ к HTML-документу, `FILE` — доступ к файлу на локальной машине, `FTP` — запрос файла с FTP-сервера и т. д.). Для указания имени ресурса, на который мы ссылаемся, используется параметр `server`. Например, для ссылки на сайт крупнейшего в Интернете продавца книг Amazon.Com используется имя www.amazon.com. После имени сервера стоит необязательный параметр `port`, который указывает номер порта, на котором функционирует тот Web-сервер, на который поставлена ссылка. По умолчанию значение параметра равно 80, и в подавляющем большинстве случаев менять его не придется.

Теперь разберемся с необязательным параметром `path`. Он указывает каталог, в котором находится ресурс. При обращении по имени Web-сервера мы попадаем на основную страницу в его корневом каталоге, а если нам надо загрузить страничку или исполняемый файл из других каталогов, то необходимо будет указать их полное имя в параметре `path`. Например, если вы на своем Web-сервере www.myserver.ru создали страничку с содержанием всего, что у вас есть в файле Content.html, который находится в каталоге Wcont, то для доступа именно к этой страничке, минуя основную страницу, необходимо написать ссылку http://www.myserver.ru/Wcont/content.html.

Мы рассмотрели внешние ссылки, которые предназначены для перехода к внешним ресурсам. Однако существуют и локальные ссылки, которые позволяют быстро переходить от одного раздела к другому внутри самого документа. Например, если у вас просматривается большой документ, состоящий из нескольких разделов, то стоит разместить в его начале перечень разделов и ссылки на каждый из них.

Для создания такой локальной ссылки необходимо сначала поставить "закладку" в то место, куда мы в последствии будем ссылаться. Это выполняется с помощью тэгов `<A NAME=имя_закладки>` и `</A>`, между которыми стоит первая строка того раздела, на который мы выставляем гиперссылку. Так мы создали саму "закладку", на которую будем ссылаться. Соответственно, сама ссылка для перехода будет выглядеть следующим образом: `<A HREF=#имя_закладки> текст самой ссылки </A>`.

Кстати, если подобные закладки определены для внешнего документа, то можно ссылаться и сразу на них. То есть, если в нашей вышеупомянутой страничке Content.html определена закладка с именем Part2, то ссылка именно на это место будет выглядеть как http://www.myserver.ru/Wcont/content.html#Part2. Также существует возможность создавать ссылки в виде графических изображений, но прежде чем мы увидим, как это делается, разберемся с самой графикой в HTML-документах.

Первое, что приходит в голову, — это создать для документа хороший фон, ибо стандартный серый фон браузеров немножко безрадостен, и, скорее всего, вам уже надоел. Эта проблема решается очень просто. Следует либо изменить цвет фона командой `<BODY BGCOLOR=#xxxxxx>` (как мы уже говорили выше), либо, если у вас есть симпатичный рисунок, который вы хотите сделать фоновым, использовать тэг `<BODY BACKGROUND="имя_файла">`. В оформлении HTML-документов могут использоваться файлы только трех графических форматов: GIF, JPEG и PNG.

Вставка графического файла в текст HTML-документа осуществляется тэгом `<IMG SRC="имя_файла">`. У этого тэга достаточно много параметров, которые позволяют добиться более точного расположения рисунка на странице, задать его ширину и высоту, рамку и многое другое.

Параметр ALT="текстовая строка" позволяет указать текстовую строку вместо графического изображения, если в браузере отключена возможность автоматической загрузки графики. Чрезвычайно рекомендован к употреблению.

Параметры HEIGHT и WIDTH задают соответственно высоту и ширину изображения в пикселах. Параметры HSPACE и VSPACE задают ширину в пикселах свободного пространства, которое отделяет графическое изображение от текста по горизонтали и вертикали соответственно.

Для выравнивания текста относительно графики (или наоборот), используется параметр ALIGN. Значения LEFT, RIGHT, TOP, BOTTOM и MIDDLE выравнивают текст относительно рисунка по левой, правой, верхней и нижней границе или по центру изображения соответственно. Значение TEXTTOP приведет к выравниванию по верхней границе относительно самых высоких символов в строке, ABSMIDDLE заставит строку текста выравниваться относительно середины изображения, ABSBOTTOM — осуществляет выравнивание нижней границы рисунка относительно нижней границы строки. Также существует возможность создавать так называемую сегментированную графику, или карту изображений, где на один рисунок приходится несколько ссылок, но это мы разберем чуть позже. А пока рассмотрим простейший пример размещения графики в тексте.

Листинг 1.4. Файл 1-4.html

```
<HTML>
 <HEAD>
  <TITLE>Моя очередная страница</TITLE>
 </HEAD>
 <BODY>
  <P>Мы пишем на <IMG SRC="html_20.GIF" ALT="HTML 2.0" HEIGHT=38
WIDTH=102 ALIGN=LEFT> </P>
 </BODY>
</HTML>
```

На рис. 1.4 мы видим результат отображения браузером файла 1-4.html.

Обратите внимание, что текст "HTML 2.0", написанный жирным шрифтом, на самом деле является рисунком, хранящимся в файле html_20.gif.

Теперь рассмотрим вопрос создания ссылок не в виде текстовых строк, а в виде графических изображений. Для этого между тэгами <A> и </A> необходимо вставить команду <IMG>. Например, для ссылки на нашу страницу с именем, которое мы придумали чуть раньше, необходимо написать строку <A HREF="http://www.myserver.ru/Wcont/content.html" <IMG>SRC="cont.gif> и закрыть ее тэгом </A>.

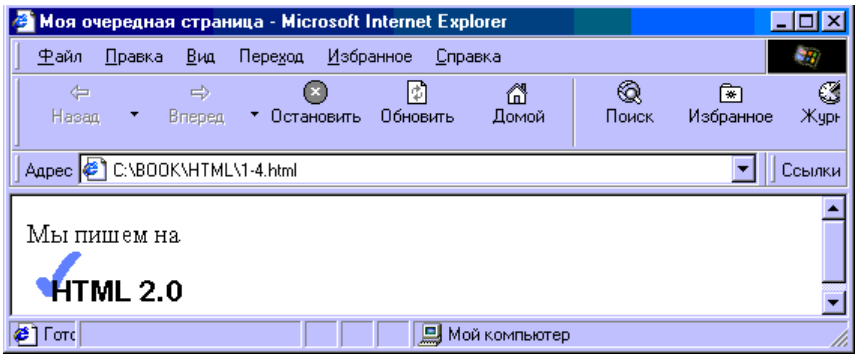

Рис. 1.4. Окно браузера с результатом просмотра файла, приведенного в листинге 1.4.

Помимо этого способа, для создания графических ссылок используется так называемая сегментированная графика, или Image map. Эта технология применяется в тех случаях, когда в одном изображении необходимо разместить несколько ссылок, переход на каждую из которых осуществлялся бы при щелчке мышью в определенном месте изображения. Например, это удобно, когда изображение меню создается в виде графического файла (по типу стандартной панели инструментов), и для каждого пункта меню реализуется своя ссылка.

При использовании сегментированной графики сначала необходимо подготовить сам графический файл. Затем он вставляется в текст HTML-документа с помощью обычного тэга вставки рисунка `<IMG>`, к которому необходимо добавить параметр `USEMAP`. В общем виде это выглядит следующим образом: `<IMG SRC="picture.gif" USEMAP="#map">`.

В этой строке `picture.gif` — файл с изображением рисунка, который мы собираемся использовать в виде карты, а `map` — имя карты чувствительных областей рисунка. Если описание карты находится не в том документе, где она помещена, то перед знаком решетки вписывается имя HTML-файла, который содержит описание этой карты.

Для создания карты областей, чувствительных к нажатию, используются тэги `<MAP NAME="map_name">` и `</MAP>`, где `map_name` — собственно имя карты. Если снова обратиться к вышеприведенному примеру, то значение параметра `NAME` будет `map`. А между открывающими и закрывающими тэгами помещаются определители чувствительных областей. Для задания каждой области применяется тэг `<AREA SHAPE="shape" COORDS="coor" HREF="link">`. Параметр `SHAPE` принимает значения `RECT`, `CIRC` и `POLY` для задания прямоугольника, круга и многоугольника соответственно. `COORDS` задает список координат для выбранной формы. Так, для прямоугольника в параметре `COORDS` через запятую перечисляются четыре координаты левого верхнего и правого нижнего угла по осям x и y соответственно. Для круга необходимо передать всего три

значения — координаты центра и радиус круга. Для многоугольника указываются пары координат каждой вершины. Ну, а параметр HREF задает URL, по которому необходимо перейти при нажатии на данную область. Так, для задания области прямоугольной формы необходимо записать приблизительно следующий тэг: <AREA SHAPE=RECT COORDS="0,0,30,30" HREF="a.html">.

Таблицы

К сожалению, HTML-документы не имеют средств для четкого расположения текста и графики на страницах. Связано это с тем, что никогда нельзя предсказать заранее, какой размер окна браузера и разрешение экрана будут выбраны пользователем. Создание же красивых и функциональных страниц требует достаточно точного позиционирования текста и рисунков. Для этих целей широко используются таблицы. Для языка HTML это на удивление гибкий инструмент. В ячейках таблицы легко размещаются и текст и графика. Однако все эти прелести требуют применения достаточно большого объема параметров. Рассмотрим их подробно в этом разделе.

В HTML-тексте таблица помещается между тэгами <TABLE> и </TABLE>. Между этими тэгами последовательно описывается каждая строка таблицы, заключаемая в тэги <TR> и </TR>, а уже внутри каждой строки описываются ячейки столбцов. Причем столбцы всегда должны находиться в обрамлении тэгов <TD> и </TD>. Например, следующая конструкция воспроизводит матрицу размером 3 на 3, состоящую из цифр от 1 до 9.

```
<TABLE>
 <TR>
  <TD> 1</TD> <TD> 2</TD> <TD> 3</TD>
 </TR>
 <TR>
  <TD> 4</TD> <TD> 5</TD> <TD> 6</TD>
 </TR>
 <TR>
  <TD> 7</TD> <TD> 8</TD> <TD> 9</TD>
 </TR>
</TABLE>
```

Впрочем, много нагляднее продемонстрировать это на примере. Чтобы таблица выглядела более презентабельно, добавим некоторые элементы оформления, например, рамку, выравнивание и ограничение размера.

Листинг 1.5. Файл 1-5.html

```
<HTML>
 <HEAD>
  <TITLE>Таблица!</TITLE>
```

```
 </HEAD>
 <BODY>
  <TABLE ALIGN=LEFT BORDER=3 WIDTH=70%>
 <TR>
  <TD> 1</TD> <TD> 2</TD> <TD> 3</TD>
 </TR>
 <TR>
  <TD> 4</TD> <TD> 5</TD> <TD> 6</TD>
 </TR>
 <TR>
  <TD> 7</TD> <TD> 8</TD> <TD> 9</TD>
 </TR>
</TABLE>
 </BODY>
</HTML>
```

Результат просмотра этого файла в браузере показан на рис. 1.5.

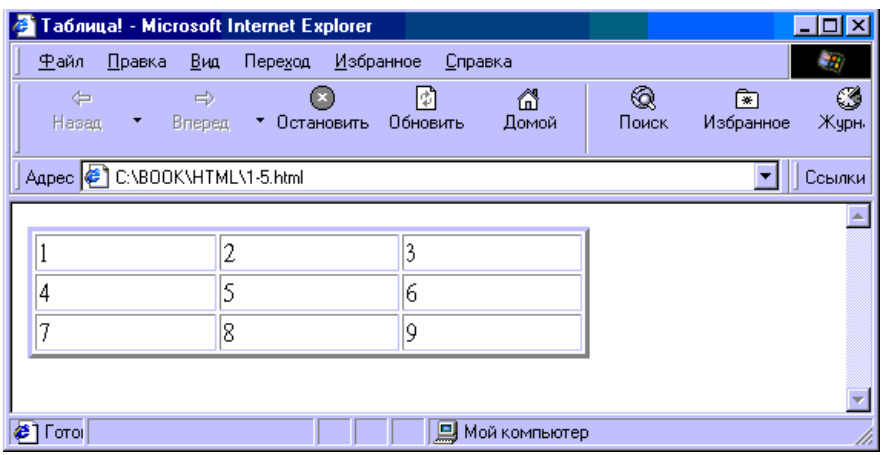

Рис. 1.5. Окно браузера с результатом просмотра файла, приведенного в листинге 1.5

Как видно, в тэге `TABLE` параметр `ALIGN` задает выравнивание таблицы по левому краю окна браузера, параметр `BORDER` задает ширину границы таблицы (в пикселах), а `WIDTH` задает ширину таблицы в пикселах, или, как в нашем случае, в процентах от ширины окна браузера.

Помимо этого, в таблицах можно размещать заголовки столбцов и строк при помощи тэгов `<TH>` и `</TH>`, а также название таблицы, заключив его между

тэгами <CAPTION> и </CAPTION>. Для иллюстрации действия этих тэгов приведен следующий пример.

Листинг 1.6. Файл 1-6.html

```
<HTML>
 <HEAD>
  <TITLE>Таблица!</TITLE>
 </HEAD>
 <BODY>
  <TABLE BORDER=3>
  <CAPTION ALIGN=bottom> Таблица №1 </CAPTION>
  <TR><TD></TD><TH>Рост</TH><TH>Вес</TH></TR>
  <TR><TD>Саша</TD><TD ALIGN=center>170</TD><TD ALIGN=center>75</TD></TR>
  <TR><TD>Маша</TD><TD ALIGN=center>168</TD><TD ALIGN=center>57</TD></TR>
  </TABLE>
 </BODY>
</HTML>
```

Результат представлен на рис. 1.6.

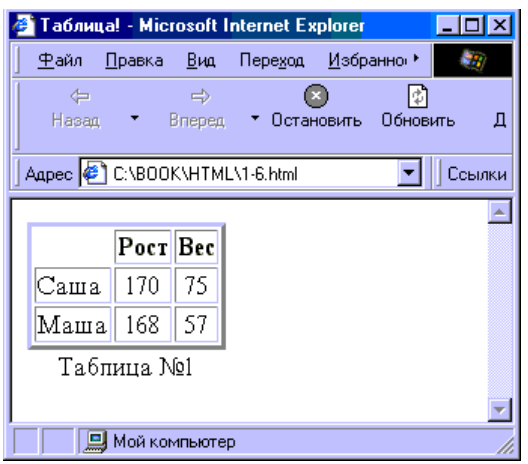

Рис. 1.6. Окно браузера с результатом просмотра файла, приведенного в листинге 1.6

Теперь рассмотрим параметры всех упомянутых тэгов. Для всех тэгов существует параметр ALIGN, принимающий значения LEFT, RIGHT, CENTER и JUSTIFY, который задает выравнивание объекта по левому или правому краю, по центру и по левому и правому краю одновременно. Кроме него, существует схожий по действию параметр VALIGN, задающий выравнивание по вертикали. Он имеет значения TOP, MIDDLE, BOTTOM и BASELINE для верти-

кального выравнивания объекта по верхней границе, по центру, по нижней границе и по базовой линии текста соответственно. Параметр BORDERCOLOR задает цвет границы объекта. Параметры BORDERCOLORDARK и BORDERCOLORLIGHT задают темный и светлый цвета для трехмерного изображения рамки вокруг объекта.

Теперь рассмотрим параметры, специфичные для каждого тэга. Начнем с тэга <TABLE>. В параметре BACKGROUND указывается имя графического файла, который будет служить фоном для всей таблицы. Если рисунок не используется, то параметром BGCOLOR можно задать цвет фона таблицы. Параметр NOWRAP запрещает перенос содержимого таблицы на другую строку, если оно не помещается в окне просмотра браузера. Параметр WIDTH, как мы уже видели, задает ширину таблицы в пикселах или процентах от ширины окна браузера. Для форматирования текста, расположенного после таблицы в теле документа, используется параметр CLEAR со значениями NO, LEFT и RIGHT. Первое значение указывает на то, что текст будет располагаться сразу после таблицы, без какого-либо выравнивания, а второе и третье — что текст будет размещен на первой строке после таблицы и будет выравниваться по левой и правой границе соответственно. Параметр COLS задает количество колонок в таблице.

Для управления внешним видом границы таблицы используется параметр FRAME. Значение BORDER, установленное по умолчанию, заставляет отображать всю рамку таблицы. Значение VOID указывает, что внешняя рамка не отображается. ABOVE, BELOW, LHS и RHS отображают верхнюю, нижнюю, левую и правую границы соответственно. Значения HSIDES и VSIDES ответственны за отображение только горизонтальных и вертикальных границ. А для установки внешнего вида границ между ячейками используется параметр RULES. Он принимает значения NONE, ROWS, COLS и ALL. Значение NONE указывает, что границы между ячейками не будут отображаться, значения ROWS и COLS отображают разделители между строками и столбцами соответственно, а значение ALL позволяет отображать разделительные линии между всеми ячейками.

По умолчанию считается, что если строки таблицы не помещаются полностью в окне браузера, то они переносятся на следующую строку, что не всегда может отвечать замыслам создателя таблицы. Параметр NOWRAP отменяет этот режим.

Все вышеперечисленные тэги работают с целой таблицей. Теперь же мы рассмотрим тэги, отвечающие за оформление отдельных ячеек. Существуют два очень специфичных тэга, используемых для операции слияния ячеек. Иногда возникает ситуация, когда необходимо объединить несколько ячеек по вертикали и горизонтали. Для этого и служат тэги COLSPAN и ROWSPAN, соответственно. В качестве примера приведем следующий листинг.

Листинг 1.7. Файл 1-7.html

```
<HTML>
 <HEAD>
  <TITLE>Таблица!</TITLE>
 </HEAD>
 <BODY>
  <TABLE BORDER=3>
  <CAPTION ALIGN=bottom> Таблица №1 </CAPTION>
  <TR><TD></TD><TH COLSPAN=2>Среднее значение</TH></TR>
  <TR><TD></TD><TH>Рост</TH><TH>Вес</TH></TR>
  <TR><TD>Александры</TD><TD ALIGN=center>170</TD><TD
ALIGN=center>75</TD></TR>
  <TR><TD>Марии</TD><TD ALIGN=center>168</TD><TD
ALIGN=center>57</TD></TR>
  </TABLE>
 </BODY>
</HTML>
```

Результат показан на рис. 1.7.

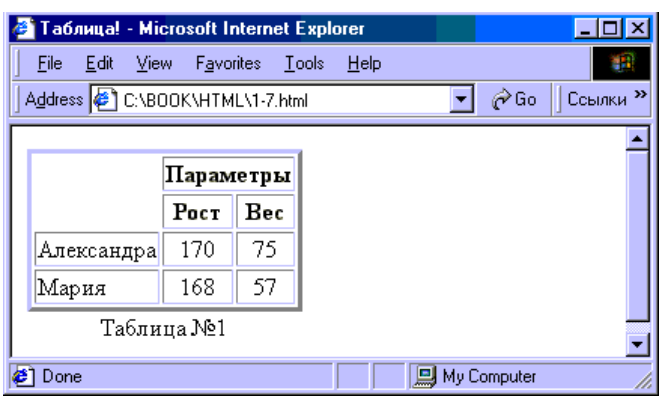

Рис. 1.7. Окно браузера с результатом просмотра файла, приведенного в листинге 1.7

Фреймы

В принципе, мы уже рассмотрели основные возможности языка HTML. Дальнейшее описание будет касаться более тонких вещей. Одна из них — структурирование HTML-документа с помощью фреймов (Frame, кадр).

На каждом сайте находится достаточно большое количество разнообразной информации. Целесообразно разносить ее по маленьким HTML-докумен-

там, чтобы свести все исправления к минимуму в случае изменения или дополнения в одном месте сайта. Фреймовая структура сайта подразумевает, что каждая страница будет состоять из отдельных областей, в каждой из которых будет отображаться один HTML-файл. В таком случае главный HTML-документ будет содержать только определения фреймов, а все наполнение фреймов разместится в других файлах.

При создании подобного документа фреймовой структуры, в главном HTML-файле нет нужды употреблять тэг <BODY>. Вместо него используются тэги <FRAMESET> и </FRAMESET>. А уже между ними вставляются тэги <FRAME>, которые и устанавливают содержимое каждого отдельно взятого фрейма.

Конкретное положение фреймов указывается в тэге <FRAMESET> с помощью параметров. Если наши фреймы расположены в виде таблицы, то используются параметры COLS и ROWS, задающие размеры колонок и строк. Ширина и высота каждого фрейма задается либо в пикселах, либо в процентах от размера окна браузера. Например, если у нас есть два фрейма, которые мы хотим расположить рядом, то используется конструкция <FRAMESET COLS="80,120">. Здесь мы указали, что в документе будет два фрейма, расположенных рядом, шириной 80 и 120 пикселов. Если вместо конкретного значения поставить знак "*", то для соответствующего фрейма будет отведено все свободное место окна браузера. Так, конструкция <FRAMESET COLS="110,*"> установит ширину первого фрейма в 110 пикселов, а для второго отведет все оставшееся место по ширине окна просмотра. Точно так же действует и параметр ROWS.

Помимо этого, в тэге <FRAMESET> используются параметры FRAMEBORDER и FRAMESPACING. Если значение первого из них равно единице, то каждый фрейм будет иметь трехмерную рамку, если нулю, то рамки не будет. Второй параметр указывает расстояние между соседними фреймами в пикселах.

Параметр FRAMEBORDER также применяется и в тэге <FRAME>. Это необходимо, если наличие рамки следует указывать для отдельных фреймов. Кроме того, в тэге <FRAME> могут быть использованы параметры MARGINHEIGHT и MARGINWIDTH для отступа по вертикали и горизонтали от границ фрейма в пикселах.

Также в тэге <FRAME> используется параметр NORESIZE. В том случае, если он указан при определении какого-либо фрейма, пользователь не сможет изменять его границы. Параметр SCROLLING задает создание полос прокрутки для просмотра содержимого фрейма. Значение YES заставляет отображать полосы прокрутки в каждом случае, значение NO, соответственно, не позволяет их отображать, а значение AUTO создает их только в случае необходимости, т. е. когда содержимое фрейма не помещается целиком в области просмотра.

И вполне естественно, что в тэге <FRAME> используется параметр SRC, указывающий на HTML-файл с содержимым данного фрейма.

Конечно, гораздо лучше все это увидеть на примере.

Листинг 1.8. Файл 1-8.html

```
<HTML>
 <HEAD>
  <TITLE>Пример с фреймами</TITLE>
 </HEAD>
 <FRAMESET COLS="40%,60%" FRAMEBORDER=1>
 <FRAME SCROLLING=AUTO SRC="1-6.html">
 <FRAME SCROLLING=AUTO SRC="1-7.html">
 </FRAMESET>
</HTML>
```

Здесь мы задаем два фрейма, шириной 40 и 60 процентов от ширины окна браузера, и указываем, какие HTML-файлы в них разместить. Результат показан на рис. 1.8.

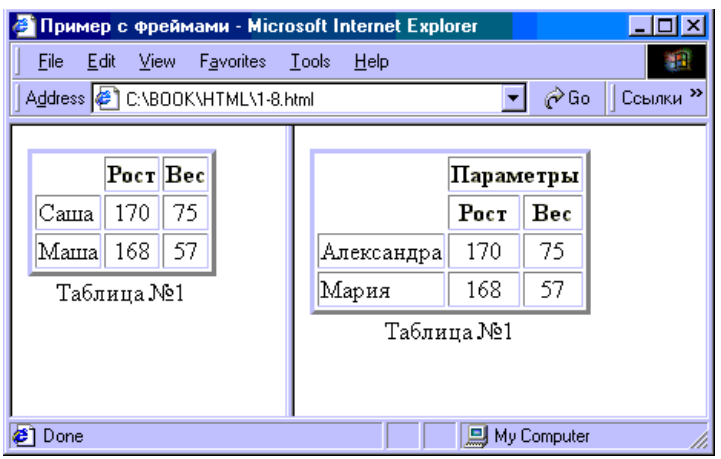

Рис. 1.8. Окно браузера с результатом просмотра файла, приведенного в листинге 1.8

Существуют еще и так называемые плавающие фреймы. Вокруг такого фрейма может располагаться текст или другое наполнение HTML-файла. Проще всего представить его в виде вставки рисунков, так как выравнивание текста относительно плавающего фрейма делается точно так же, как и выравнивание относительно графического изображения. Плавающий фрейм, задаваемый тэгами `<IFRAME>` и `</IFRAME>`, имеет параметр `ALIGN`, идентичный такому же параметру для тэга размещения графики `<IMG>`, и параметры

WIDTH, HEIGHT, HSPACE и VSPACE, чьи значения и смысл также совпадают с уже известными нам параметрами тэга <IMG>.

Разницу между обычными и плавающими фреймами можно объяснить следующим способом. Если мы задаем стандартные фреймы, то они полностью определяют всю структуру данного HTML-документа. Поместить какое-либо содержимое вне фреймов, как минимум, проблематично, ведь во фреймовом документе нет тэга <BODY>. А плавающий фрейм позволяет органично вставлять его в тело обычного документа вместе с графикой и текстом.

В качестве примера приведем маленький HTML-документ, в тело которого вставлен плавающий фрейм с содержимым файла 1-3.html.

Листинг 1.9. Файл 1-9.html

```
<HTML>
 <HEAD>
 <TITLE>Плавающий фрейм</TITLE>
  <BODY>
  Размещение плавающих фреймов
   <IFRAME SRC="1-3.html" ALIGN=RIGHT HEIGHT="50%" WIDTH="70%"
SCROLLING=AUTO> </IFRAME>
   Обычный текст документа, который может быть расположен рядом с фреймом
  </BODY>
 </HEAD>
</HTML>
```

Результат представлен на рис. 1.9.

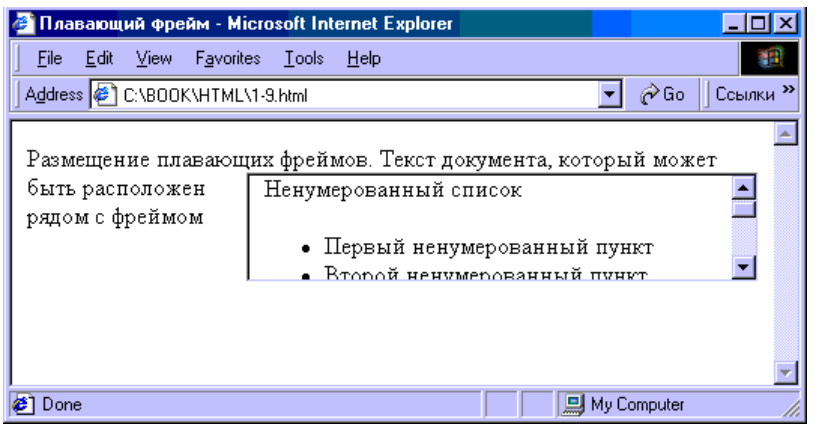

Рис. 1.9. Окно браузера с результатом просмотра файла, приведенного в листинге 1.9

Однако, к сожалению, плавающие фреймы распознает только Internet Explorer от Microsoft. Netscape Navigator попросту игнорирует эти конструкции. Поэтому к применению плавающих фреймов надо относиться очень осторожно, и использовать только в том случае, когда вы твердо убеждены, что ваше творение будет просматриваться только при помощи браузера Internet Explorer (например, в корпоративной сети intranet, где у всех пользователей одинаковое программное обеспечение).

Не исключен вариант, когда HTML-документ будет просматриваться при помощи браузера, у которого вообще нет возможностей для работы с фреймами. В таком случае после определения фреймовой структуры документа после тэга `</FRAMESET>` вставляется пара тэгов `<BODY>` и `</BODY>`. А уже между ними помещается второй вариант содержимого странички, рассчитанный на тех несчастных пользователей, чьи браузеры не имеют возможностей для просмотра HTML-документов с фреймами.

Формы

В этой главе мы рассмотрим создание активных форм с помощью языка HTML. Это, пожалуй, наиболее близко именно к программированию. К сожалению, возможности HTML для этих целей достаточно скудны, поэтому для серьезных вещей их не хватит. Отметим, что все эти возможности почти никогда не используются сами по себе. Обычно они дополняются приложениями CGI. Давайте немного разберемся в теории действия этого механизма.

Язык HTML применяется для создания страниц, которые просто несут информацию, т. е. не изменяются динамически. Они статичны. Если же мы хотим получить интерактивные сайты, то должны прибегнуть к дополнительным средствам. Существует метод, который позволяет принимать разнородную информацию с компьютера пользователя. Для его реализации и используются приложения CGI. Что же такое CGI? CGI (Common Gateway Interface, общедоступный шлюзовый интерфейс) — это интерфейс для запуска внешних программ на самом сервере. Пользуясь этим интерфейсом, приложения CGI могут получать информацию от удаленного пользователя, анализировать ее, формировать новый HTML-документ и отправлять его обратно пользователю.

Например, удаленный пользователь хочет провести сортировку таблицы, которую он просматривает. Он вводит параметр сортировки в HTML-документ на своем удаленном компьютере. Затем, при нажатии на соответствующую кнопку, эти данные посылаются серверу, а конкретно, CGI-приложению, запущенному на нем. Оно интерпретирует полученные данные, проводит сортировку таблицы, формирует новый HTML-документ с уже отсортированной таблицей и передает его удаленному пользователю. С его же стороны все это выглядит как обычная работа со стандартным

Windows-приложением: ввел параметры в строку ввода, нажал на кнопку, и таблица тут же отсортировалась. Но теперь мы знаем, что вся работа была проделана сервером, а пользователь просто получил новый документ.

Однако прежде чем мы начнем изучать CGI-приложения, познакомимся все-таки с возможностями самого языка HTML.

До того, как мы начнем рассматривать CGI-приложения, сначала все-таки ознакомимся с возможностями самого языка HTML.

Итак, в HTML-документ форма вставляется при помощи тэгов `<FORM>` и `</FORM>`, между которыми помещаются тэги, определяющие элементы управления. У тэга `<FORM>` есть несколько параметров. Прежде всего, это параметр `ACTION`, который указывает URL того CGI-приложения, которое будет взаимодействовать с нашей формой. Параметр `METHOD` указывает метод, который будет использоваться для передачи данных из формы в CGI-приложение. Этот параметр может принимать два значения: `GET` и `POST`. Разница между ними состоит в том, что при значении `GET` вся информация помещается в переменную среды с именем `QUERY_STRING`, а при значении `POST` CGI-программа получит все данные через стандартный поток ввода. Поскольку в этом случае мы не можем заранее сказать, сколько информации будет передано, то используем переменную среды `CONTENT_LENGTH` для того чтобы задать количество информации, которое необходимо считать из стандартного потока ввода.

А теперь разберемся с самими элементами управления. Каждый такой элемент вставляется с помощью тэга `<INPUT>`. Параметр `TYPE` задает тип элемента. Так, значение `RADIO` создает переключатели, то есть элементы, зависимые друг от друга. А значение `CHECKBOX` предназначено как раз для создания флажков. `TEXT` используется для ввода текстовой информации. Размеры определяются дополнительными параметрами `SIZE` и `MAXLENGTH`. Параметр `PASSWORD` практически идентичен органу управления `TEXT`, но предназначен для ввода пароля, и соответственно, отображает вводимую информацию в скрытом виде, заменяя содержимое звездочками. Помимо этого, существует несколько типов кнопок. Обычная кнопка задается значением `BUTTON`. Кнопка, заданная значением `RESET`, сбрасывает все введенные данные во всех органах управления и возвращает их к начальному состоянию, заданному с помощью параметра `VALUE` (его мы рассмотрим чуть позже). Кнопка типа `SUBMIT` предназначена для посылки всех данных из заполненной формы в CGI-приложение. Эта кнопка обычно вставляется в конце формы. Впрочем, помимо тривиальной кнопки для отправки данных можно использовать графическое изображение. Для этого параметру `TYPE` приписывается значение `IMAGE`. Все остальные данные необходимо указывать как при вставке обычного графического изображения.

Это были возможные значения параметра `TYPE` для тэга `<FORM>`. У этого тэга существует несколько параметров. Обязательный параметр `NAME` задает имя

элемента управления, которое будет посылаться программе-обработчику. Параметр VALUE задает начальное значение элемента управления (например, текст по умолчанию для строки текстового ввода). Уже упоминавшийся параметр SIZE указывает ширину поля ввода для текстовых элементов управления, а MAXLENGTH — максимальное количество символов, которое можно ввести в это поле. Для различных переключателей (RADIO) и флажков (CHECKBOX) параметр CHECKED указывает, что данный элемент необходимо отметить как включенный. Кстати, для этой пары управляющих элементов есть одна особенность в применении параметра NAME. Она состоит в следующем. Для каждого флажка устанавливается свое значение параметра NAME, а для каждого переключателя (радиокнопки) одной группы значение NAME должно быть одинаковым. Параметр ALIGN нам уже достаточно знаком, и, как можно догадаться, ответственен за выравнивание текста, расположенного около формы. Для графического изображения (TYPE=IMAGE) используется параметр SRC, который указывает расположение графического файла.

Помимо тэга <INPUT>, можно использовать тэг <SELECT>, который применяется для создания выпадающих списков. Он имеет два параметра — NAME и SIZE, которые задают имя и высоту списка в строках. Вполне естественно, что для оператора <SELECT> существует и закрывающий тэг </SELECT>. Между ними и помещаются строки, входящие в выпадающий список выбора, каждая из которых предваряется тэгом <OPTION>. Для отметки строки, выбираемой по умолчанию, используется <OPTION SELECTED>. А теперь рассмотрим все это на примере.

```
<SELECT NAME="spisok">
<OPTION SELECTED> Первая строка
<OPTION> Вторая строка
<OPTION> Третья строка
</SELECT>
```

В этом примере создан выпадающий список из трех строк, в котором первая строка выбрана по умолчанию.

И в этой же группе тэгов мы должны отметить тэг <TEXTAREA> с его закрывающей парой </TEXTAREA>, которые создают поле для ввода многострочного текста. Естественно, у этого тэга есть параметр NAME, и еще добавлены параметры ROWS и COLS, задающие размер поля ввода по вертикали и горизонтали в строках и символах соответственно.

Посмотрим это все на примере.

Листинг 1.10. Файл 1-10.html

```
<HTML>
 <HEAD>
```