КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА ЭКОНОМИЧЕСКОЙ КИБЕРНЕТИКИ

ПИНЯГИНА О.В.

Разработка электронного магазина на PHP и MySQL

Казань – 2010

УДК 004.738.5
ББК 32.973.26-018.02

Печатается по постановлению редакционно-издательского совета
факультета вычислительной математики и кибернетики
Казанского государственного университета

Рецензенты:
……..
…….

Пинягина О.В. Разработка электронного магазина на PHP и MySQL/ О.В. Пинягина – Казань: Казанский государственный университет, 2010. – 108 с.

Данное учебно-методическое пособие разработано в поддержку курса "**Электронная коммерция**" и предназначено для проведения компьютерных лабораторных занятий и для самостоятельной работы студентов, обучающихся по специальности "**Бизнес-информатика**".

В пособии поэтапно рассматривается процесс создания электронного магазина на базе web-сервера Apache, языка серверных сценариев PHP и СУБД MySQL.

Электронный ресурс по данному курсу находится на сайте кафедры экономической кибернетики Казанского государственного университета по адресу: **http://kek.ksu.ru/EOS/ITE/index.html**

Содержание

Предисловие

В узком смысле **электронную коммерцию** обычно представляют себе просто как торговлю через Интернет. В самом широком смысле электронную коммерцию следует понимать как **электронный бизнес**, т.е., любой вид деятельности, в той или иной степени использующий информационные системы и глобальные сети передачи данных. Но так или иначе, **электронные магазины** – это то, с чего начала развиваться электронная коммерция.

В данном учебном пособии подробно и поэтапно рассматривается процесс разработки книжного электронного магазина на базе web-сервера **Apache**, языка серверных сценариев **PHP** и системы управления базами данных **MySQL**. Процесс разработки магазина включает следующие этапы:

1. разработка архитектуры электронного магазина;
2. разработка структуры базы данных;
3. разработка шаблонов и главной страницы;
4. разработка витрины магазина;
5. разработка корзины покупателя;
6. разработка системы регистрации и авторизации посетителей;
7. разработка системы заказа.

От читателя не требуется предварительных знаний по программированию для Интернет. В пособии имеется отдельная глава, посвященная основам языка гипертекстовой разметки **HTML**. Основная часть учебного пособия посвящена изучению серверной технологии **PHP**.

С другой стороны, предполагается, что читатель прослушал курс "Базы данных" и имеет представление о проектировании баз данных, системах управления базами данных и языке **SQL**. В пособии эти вопросы рассматриваются очень кратко – только в той степени, которая необходима для текущей задачи создания электронного магазина.

Этап 1. Разработка архитектуры электронного магазина

Разработка любого проекта начинается с формулировки требований. Итак, сформулируем в произвольной форме постановку нашей задачи.

Требуется создать электронный книжный магазин, в котором потенциальные покупатели могли бы просматривать, выбирать и заказывать книги.

Книги должны быть сгруппированы по двум признакам – по издательству и по категории (компьютерная литература, художественная литература, справочники и т. п.). Для каждой книги должна быть представлена информация о названии, авторе, цене, количестве страниц, а также внешний вид обложки.

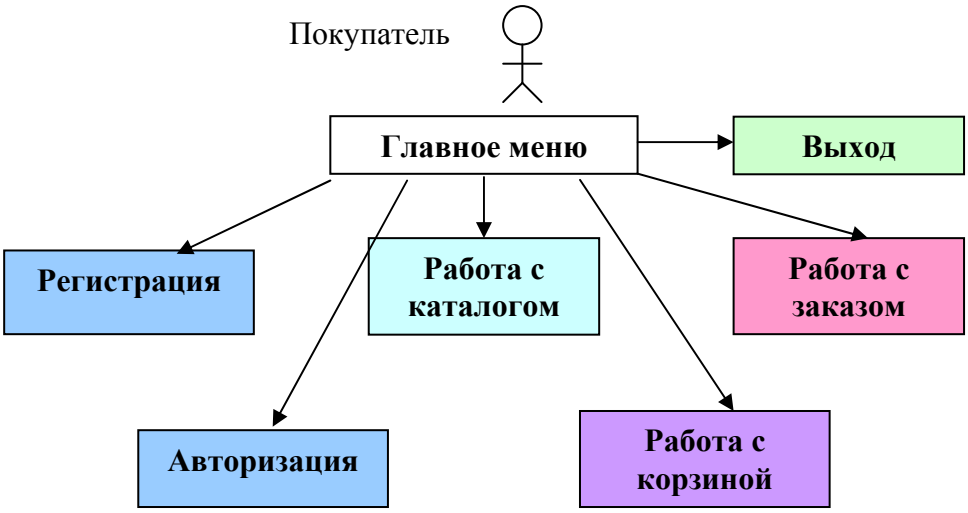
В процессе просмотра каталога посетители должны иметь возможность отложить понравившуюся книгу в корзину. При этом не требуется предварительной регистрации. Книги в корзину можно добавлять, удалять и изменять их количество.

Перед тем как оформить заказ, покупатель должен зарегистрироваться, т.е., заполнить форму с личной информацией. Эта форма должна содержать фамилию, имя, адрес, электронный адрес, логин и пароль. Предполагается, что логины являются уникальными. Информация о зарегистрированных покупателях хранится, и при следующем посещении магазина покупателю будет достаточно только авторизоваться, т.е., набрать свой логин и пароль.

Итак, для оформления заказа покупателю следует авторизоваться, после этого список выбранных книг можно просмотреть еще раз и подтвердить заказ. При этом корзина должна очищаться.

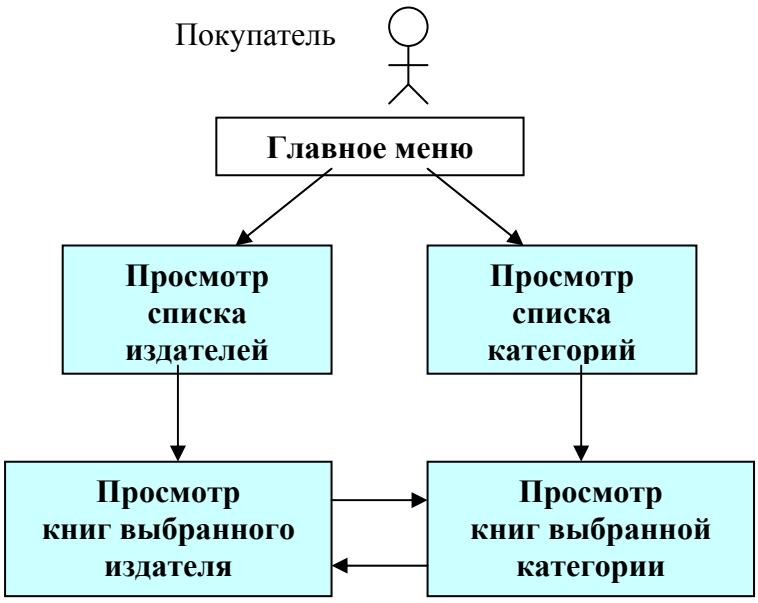
Таким образом, в нашем электронном магазине предполагается единственный тип пользователей – покупатель. Общую архитектуру сайта представим на схеме 1, где отражены основные режимы работы пользователя. Более подробно каждый режим представлен на схемах 2-6.

Схема 1. Архитектура сайта – основные режимы работы



При работе с каталогом должна иметься возможность просмотра списка издателей и списка категорий (оба режима можно реализовать в пределах одной страницы). Каждое название издателя и каждая категория будут представлять собой ссылку, при щелчке по которой будет выведен список книг выбранного издателя или выбранной категории.

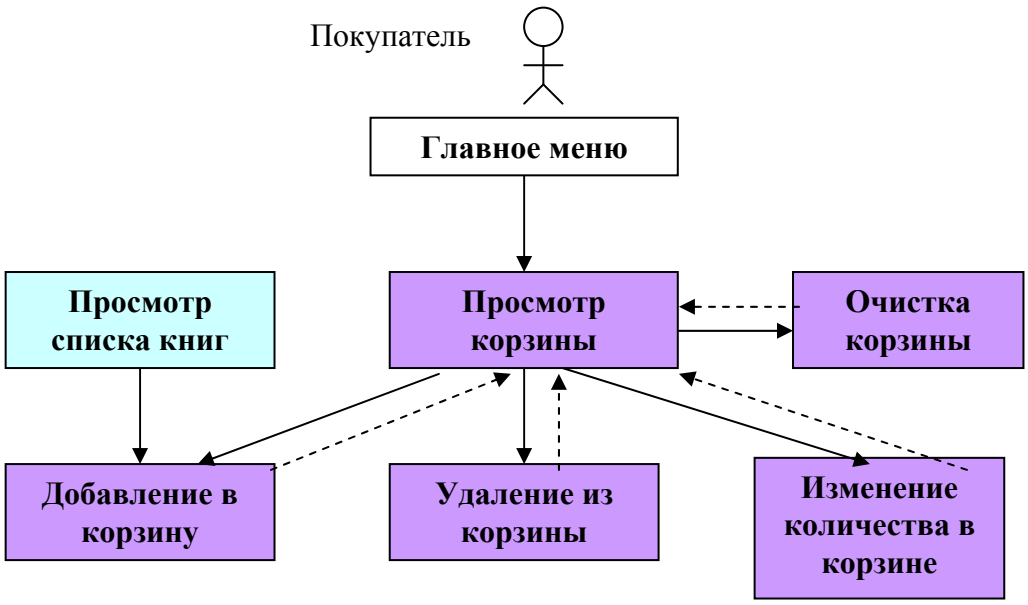
Схема 2. Работа с каталогом



При просмотре каталога рядом с каждой книгой должна быть гиперссылка или кнопка "**Положить в корзину**". При щелчке на ней книга добавляется в корзину и на экран выдается состав корзины. В корзине рядом

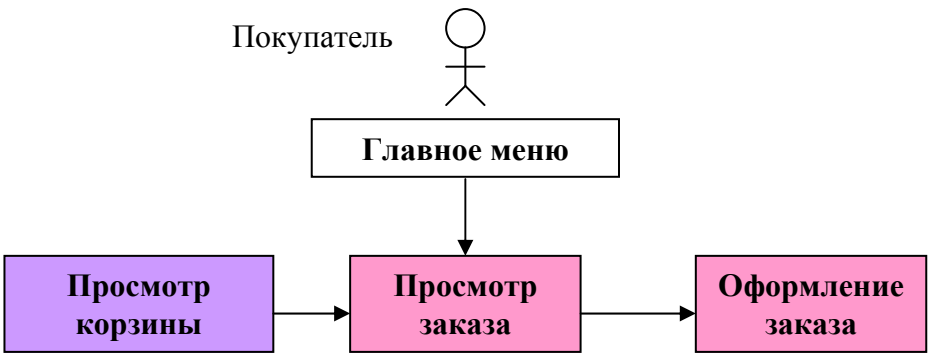
с каждой книгой выводится ее количество и кнопки или гиперссылки с надписями "**Увеличить количество**", "**Уменьшить количество**" и "**Удалить**". Кроме того, должна быть гиперссылка "**Очистить корзину**". После выполнения действий с корзиной происходит автоматический возврат в режим просмотра корзины – на схеме это изображено пунктирными стрелками.

Схема 3. Работа с корзиной



К работе с заказом можно перейти как из главного меню, так и из корзины. При этом на экран выдается состав заказа (уже без возможности редактирования) и имеется кнопка или гиперссылка "**Оформить заказ**".

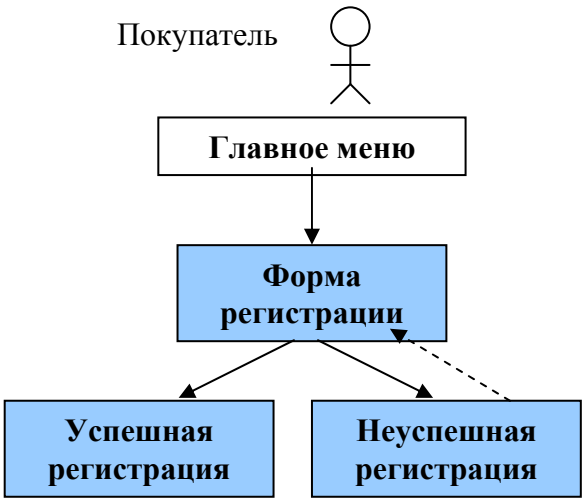
Схема 4. Работа с заказом



Для регистрации новых покупателей из главного меню можно вызвать форму регистрации. После заполнения полей формы должна быть

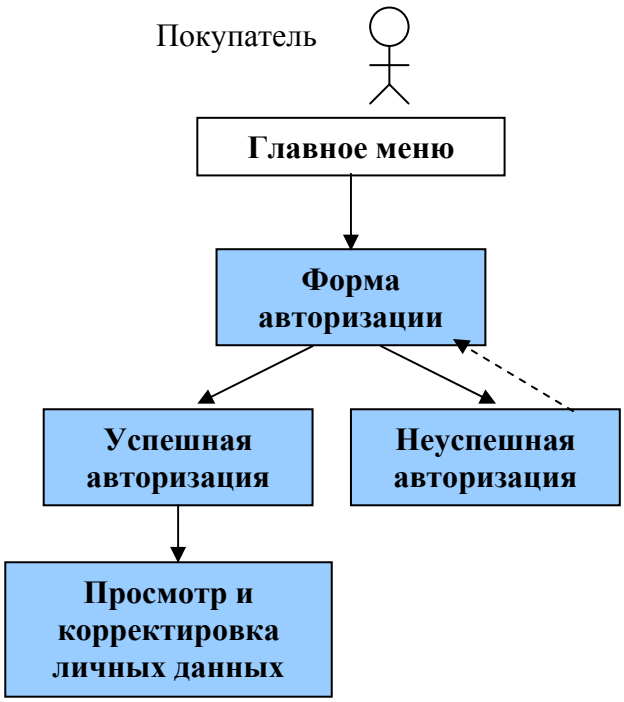
произведена проверка правильности – все ли обязательные поля заполнены, нет ли уже в базе данных пользователя с таким логином, и т.п. В случае неверно введенных данных пользователю следует выдать сообщение и снова вывести на экран форму (возможно, уже частично заполненную).

Схема 5. Регистрация

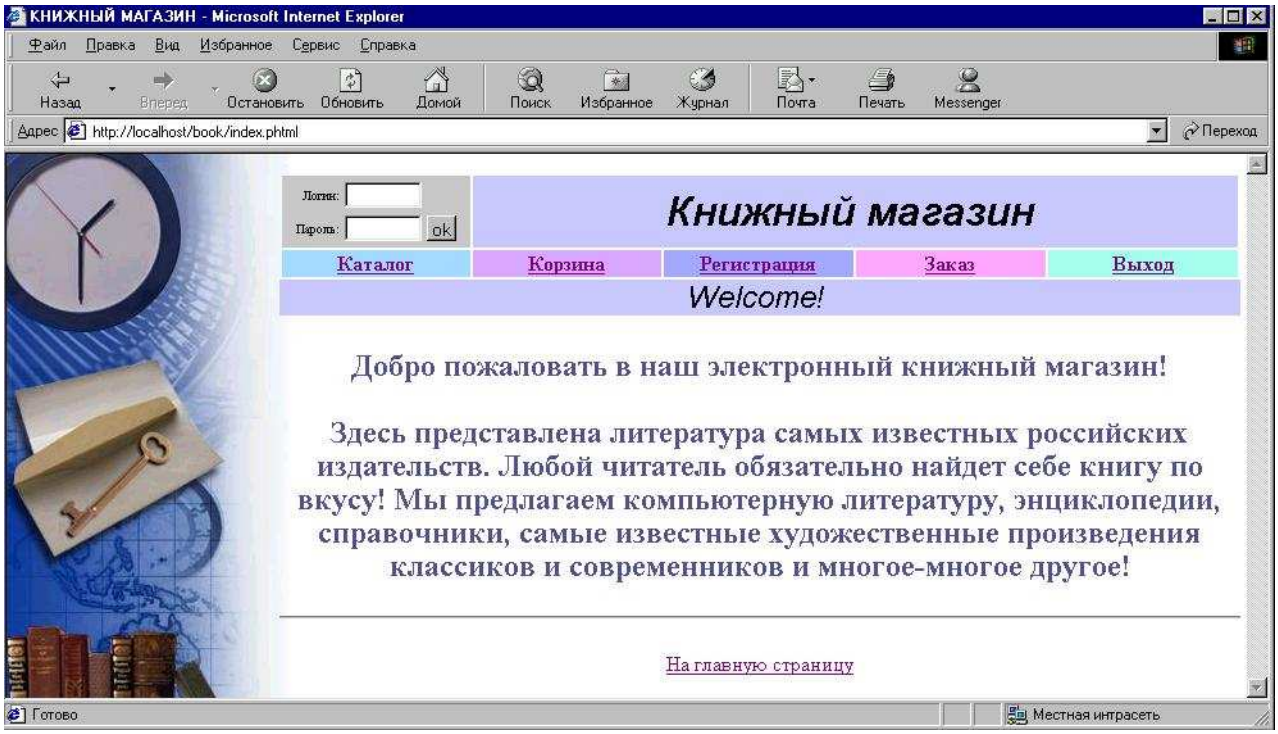


Для пользователей, которые уже зарегистрированы в нашем магазине, должна быть предусмотрена возможность авторизации, т.е., входа в магазин по логину и паролю. Если введены неверный логин и/или пароль, следует сообщить об этом. Если логин и пароль верные, пользователю следует вывести на экран его данные (ФИО, адрес, e-mail и т.п.) для возможной корректировки, а также список его предыдущих заказов.

Схема 6. Авторизация



Главная страница нашего электронного магазина может иметь примерно такой вид:



HTML-код этой страницы выглядит таким образом:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
     <title>КНИЖНЫЙ МАГАЗИН</title>
</head>

<body background="EULA.jpg" style="background-repeat:repeat-y"
leftmargin="130" rightmargin="5" bgProperties=fixed>

<table border="0" align="right" width="90%" cellpadding="0"
cellspacing="0">
<tr><td>

<table border="0" align="right" width="100%" >
<tr>
<td align="center" bgcolor="#cccccc">
<form action="auto.phtml" method="post">
<table>
<tr><td align="right"><font size=-2>Логин:</font></td>
<td align="left"><input type=text style="width:60; height:20;"
name=login></td></tr>
<tr><td align="right"><font size=-2>Пароль:</font></td>
```

```
<td align="left"><input type=password style="width:60;
height:20;" name=pass>
<input type=submit value=ok style="height:20;"></td></tr>
</table>
</td>
</form>
<td colspan="4" align="center" bgcolor="#ccccff">
<font face="Arial" size="+3"><i><b>Книжный
магазин</b></i></font></td></tr>
<tr><td align="center" bgcolor="#aaddff" width="20%">
<a href="catalog.phtml"><b>Каталог</b></a></td>
<td align="center" bgcolor="#ddaaff" width="20%">
<a href="basket.phtml"><b>Корзина</b></a></td>
<td align="center" bgcolor="#aaaaff" width="20%">
<a href="reg.phtml"><b>Регистрация</b></a></td>
<td align="center" bgcolor="#ffaaff" width="20%">
<a href="order.phtml"><b>Заказ</b></a></td>
<td align="center" bgcolor="#aaffee" width="20%">
<a href="exit.phtml"><b>Выход</b></a></td>
</tr>
</table>
</td></tr>
<tr><td align="center">Welcome!<font face="Arial"
size="+2"><i></i></font><br>
</td></tr>

<tr><td>
<center><h2><font color="#555599"><br>Добро пожаловать в наш
электронный книжный магазин! <br><br>Здесь представлена
литература самых известных российских издательств. Любой
читатель обязательно найдет себе книгу по вкусу! Мы предлагаем
компьютерную литературу, энциклопедии, справочники, самые
известные художественные произведения классиков и современников
и многое-многое другое!</font></h2></center>
</td></tr>
<tr><td><center><hr><br>
<a href="index.phtml">На главную страницу</a></center></td></tr>

</table>
</body>
</html>
```

Для создания web-страниц нам не обойтись без знания языка **HTML**, основы которого и будут рассмотрены в следующей главе.

Основы HTML

Основные тэги

HTML (HyperText Markup Language) – это язык гипертекстовой разметки, на котором написаны практически все страницы Всемирной паутины. Команды этого языка называются **тэгами** и представляют собой ключевые слова в угловых скобках. Регистр букв не имеет значения. Большинство команд являются контейнерами, имеющими открывающий и закрывающий тэги. Базовая структура HTML-документа представлена следующими командами:

```
<html>
<header>
<title>Заголовок окна</title>
</header>
<body>Содержимое документа</body>
</html>
```

Как видно, тэг `<html></html>` является тэгом самого верхнего уровня в документе. Внутри него имеются заголовочная часть (`<header>`, содержащая, в частности, строку заголовка документа `<title>`) и тело документа (`<body>`).

Язык **HTML**, как следует из его названия, предназначен прежде всего для **разметки**, или **форматирования** документа. Приведем несколько примеров команд форматирования:

Жирный шрифт	**Жирный** шрифт
курсив	*Курсив*
<u>подчеркнутый</u> шрифт	подчеркнутый шрифт
верхний индекс: A_i	верхний индекс: A^{i}
нижний индекс: Aⁱ	нижний индекс: A_{i}
центрирование	центрирование

Для переноса на следующую строку используется команда `<br>` (без закрывающего тэга).

Для рисования горизонтальной линии – команда `<hr>` (также без закрывающего тэга).

Для задания заголовков различного размера существуют шесть команд: от <h1></h1> (самый крупный) до <h6></h6> (самый мелкий).

Для создания маркированных и нумерованных списков используются следующие команды:

   понедельник вторник среда 	• понедельник • вторник • среда
   понедельник вторник среда 	1. понедельник 2. вторник 3. среда

Многие тэги могут содержать внутри себя дополнительные параметры. Параметр имеет имя и (почти всегда) значение. Так, например, тег

```
<body bgcolor="red">
```

задает цвет фона документа (красный), а тег

```
<body background="EULA.jpg">
```

задает графический файл, который будет использован в качестве фона для документа (в данном примере предполагается, что графический файл расположен в том же каталоге, что и **HTML**-документ).

Параметров внутри тэга может быть несколько. В примере

```
<body background="EULA.jpg" leftmargin="130" rightmargin="5"
bgProperties=fixed>
```

параметры `leftmargin="130"` и `rightmargin="5"` задают отступ тела документа от границ рабочей области слева и справа (в пикселах), а параметр `bgProperties=fixed` фиксирует фон документа, т.е., заставляет его быть неподвижным при прокрутке документа.

Другой пример с несколькими параметрами: тег

```
<font face="Arial" size="+2" color="green">
```

будет выводить заданный текст зеленым цветом, шрифтом Arial и размером на 2 больше, чем текущий шрифт.

Кстати, для задания цвета удобно использовать формат **#rrggbb**, где rr, gg и bb – это двузначные шестнадцатеричные числа, представляющие собой интенсивность красной, зеленой и синей компонент цвета, соответственно. Например, #000000 – черный цвет, #ffffff – белый цвет, #ff0000 – ярко-красный, #880088 – умеренно-фиолетовый (смесь красного и синего).

Для размещения картинок используется тег `<img>`, например,

```
<img src="Logotype.gif" alt="Наш логотип" width=300 height=300>
```

Параметр **src** задает имя графического файла, в качестве имени может быть использован и HTTP-адрес, например:

```
<img src="http://kek.ksu.ru/EOS/ITE/Logotype.gif">
```

Параметр **alt** задает альтернативный текст, который будет выводиться при наведении курсора мыши на картинку.

Самой полезной командой, которая и позволила логически связать разрозненные документы в единую Всемирную паутину, является тег `<a>` "гиперссылка". Например, команда

```
<a href="http://kek.ksu.ru/index.html">Кафедра экономической
кибернетики</a>
```

создает гиперссылку Кафедра экономической кибернетики, которая выглядит как подчеркнутый текст синего (по умолчанию) цвета, при щелчке мышью по ней загружается страница с адресом, заданным в параметре **href**. Если требуется загрузить страницу из текущего каталога, то можно указывать только имя файла, например:

```
<a href="basket.phtml">Корзина</a>
```

В этом тэге можно также передавать параметры в вызываемую страницу следующим образом:
href="имяФайла?параметр1=значение1&параметр2=значение2" , например,

```
<a href="basket.phtml?type=1&id_Book=25">Положить в корзину</a>
```

Эти параметры можно использовать в вызываемой странице. Более подробно о передаче параметров мы будем говорить при изучении языка серверных сценариев **PHP**.

Таблицы

Как вы уже поняли, текст и другие элементы страницы выводятся в **HTML**-документ сплошным потоком, слева направо, сверху вниз. Для того чтобы упорядочить элементы страницы друг относительно друга, самым удобным способом являются таблицы (обрамление ячеек можно убрать).

Таблица создается тэгом `<table></table>`. Полезными параметрами для таблицы являются:

- `border` – ширина рамки (если 0, то рамки нет),
- `align` – выравнивание самой таблицы по горизонтали (`"right"`, `"left"` или `"center"`),
- `width` – ширина таблицы в процентах относительно ширины страницы (`"100%"`) или в пикселах (`"500"`),
- `bgcolor` – цвет фона в таблице.

Например, таблица без рамки, шириной в 80% страницы и выравниванием по правому краю может быть задана так:

```
<table border="0" align="right" width="80%">
```

Для создания строки внутри таблицы используется тэг `<tr></tr>`.

Для создания ячейки внутри строки используется тэг `<td></td>`.

Выравнивание содержимого ячейки, ширину ячейки и цвет фона можно задавать теми же параметрами `align`, `width` и `bgcolor`. Кроме того, иногда бывает удобно объединить несколько ячеек по горизонтали или вертикали: например, параметр `colspan=3` объединяет 3 ячейки по горизонтали, `rowspan=2` объединяет 2 ячейки по вертикали.

Таким образом, например, таблица

Расписание	
Понедельник	8:00-12:00
Вторник	12:00-16:00
Среда	8:00-12:00
Четверг	8:00-12:00
Пятница	12:00-16:00
Суббота	9:00-13:00
Воскресенье	выходной

может быть создана с помощью следующего кода

```
<TABLE BORDER=1 WIDTH=100%>
<TR><TD COLSPAN=2 ALIGN="CENTER">Расписание</TD>
</TR>
<TR><TD ALIGN="RIGHT" WIDTH="50%">Понедельник</TD>
<TD WIDTH="50%">8:00-12:00</TD></TR>
<TR><TD ALIGN="RIGHT">Вторник</TD>
<TD>12:00-16:00</TD></TR>
<TR><TD ALIGN="RIGHT">Среда</TD>
<TD>8:00-12:00</TD></TR>
<TR><TD ALIGN="RIGHT">Четверг</TD>
<TD>8:00-12:00</TD></TR>
<TR><TD ALIGN="RIGHT">Пятница</TD>
<TD>12:00-16:00</TD></TR>
<TR><TD ALIGN="RIGHT">Суббота</TD>
<TD>9:00-13:00</TD></TR>
<TR><TD ALIGN="RIGHT">Воскресенье</TD>
<TD>выходной</TD></TR>
</TABLE>
```

Формы

Иногда web-страницам следует передавать параметры: например, логин и пароль при авторизации, номер книги при добавлении ее в корзину и т.п.
Как уже говорилось, параметры можно передавать непосредственно в гиперссылке после имени файла через знак вопроса. Но этот способ не всегда удобен: во-первых, потому что эти параметры будут отображаться в адресной строке браузера вместе с именем файла, а для пароля это недопустимо; а во-вторых, потому что объем информации при этом ограничен. И вообще, через адресную строку не все типы данных можно передать – например, таким образом нельзя переслать целый файл.

Поэтому в языке **HTML** существует такое понятие, как форма.

Форма – это контейнер (`<form> </form>`), содержащий элементы управления (текстовые поля, кнопки, флажки, радиокнопки, списки). Форма имеет следующие полезные параметры:

- `method` – способ (`GET` или `POST`), которым данные передаются не сервер (при использовании метода GET параметры передаются в адресной строке, как и в гиперссылке, а при использовании метода POST параметры передаются в теле запроса);
- `action` – файл, который будет загружаться как реакция на нажатие кнопки типа **Submit** (об этой кнопке см. ниже).

Внутри контейнера могут располагаться следующие элементы. Каждый элемент должен иметь имя.

Текстовые поля:

`<input type="text" name="login">` - обычное текстовое поле;

`<input type="password" name="pass">` - поле для ввода пароля, при вводе символов в это поле на экране будут видны только "звездочки".

`<input type="hidden" name="id_book">` - скрытое поле, на экране его не видно, оно полезно для передачи служебной информации, которую пользователю видеть не следует.

Кнопки:

`<input type="button" name="mybutton" value="OK">` - обычная кнопка;

`<input type="reset" name="reset" value="очистить">` - кнопка, очищающая все поля формы;

`<input type="submit" name="submit" value="отправить">` - кнопка для отправки данных на сервер: при нажатии этой кнопки вызывается web-страница, имя которой задано в параметре `action` формы, при этом на сервер передаются значения элементов управления формы.

Флажки:

```
<input type="checkbox" name="cCard" value="1"> Кредитная карта –
```

независимый переключатель, имеющий два состояния – "включен" и "выключен". Если переключатель выключен, то его значение на сервер передаваться не будет. Для того чтобы включить флажок по умолчанию, следует указать атрибут `checked`.

Радиокнопки:

```
<input type="radio" name="rCard" value="1" checked >Visa
<input type="radio" name="rCard" value="2">MasterCard
<input type="radio" name="rCard" value="3">American Express
```

- зависимые переключатели. Для того чтобы зависимые переключатели рассматривались, как единая группа, следует называть их одним именем. Если указан атрибут `checked`, то эта радиокнопка будет выбрана по умолчанию.

Списки:

```
<select size="1" name="sCard">
<option value='1' selected > Visa </option>
<option value='2'> MasterCard </option>
<option value='3'> American Express </option>
</select>
```

- обычный выпадающий список из 3-х элементов. Значение по умолчанию отмечено атрибутом `selected`.

Каким образом переданные данные анализируются на сервере, мы будем подробно рассматривать при изучении серверной технологии **PHP**.

Итак, мы рассмотрели все тэги и их параметры, которые используются в нашем учебном пособии. Единственное исключение – параметр `style`, который служит для более тонкой настройки стиля. Рассмотрение стилей выходит за рамки нашей книги. Заинтересованный читатель может обратиться к специальной литературе по **CSS** (каскадным таблицам стилей). В частности, стили рассматриваются в электронном учебнике по Web-программированию (http://kek.ksu.ru/EOS/TESTS/index.html)

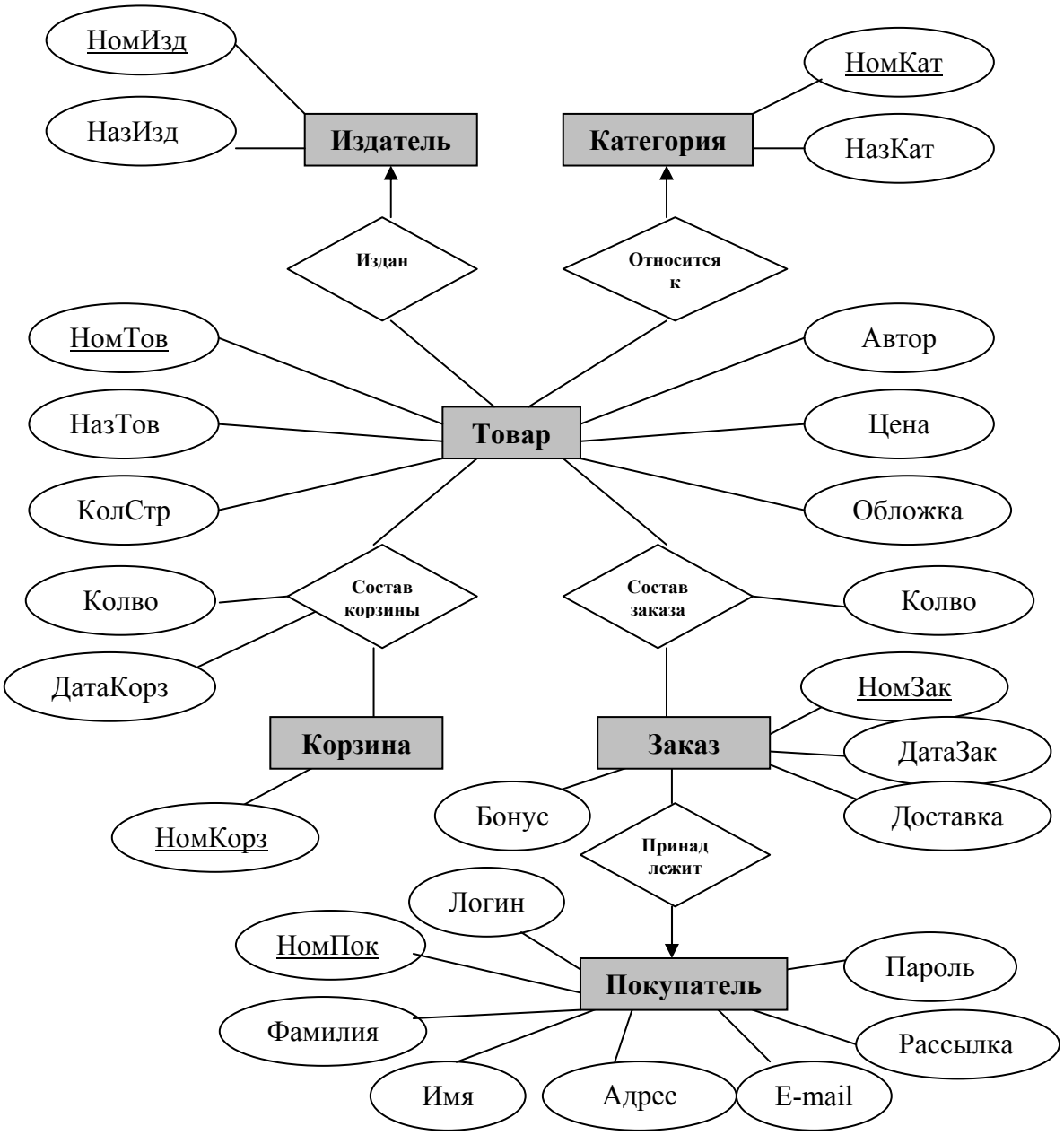
Этап 2. Разработка структуры базы данных

ER-модель

На основе описания требований к электронному магазину составим модель сущностей–связей для базы данных.

Центральным объектом в нашей модели данных является **товар** (книги). Каждая книга, кроме простых атрибутов, характеризуется связями с объектами **издатель** и **категория**. Тип связей – "многие к одному".

Для каждого потенциального покупателя создается анонимный объект **корзина,** который характеризуется только уникальным номером. Между объектом **корзина** и объектом **товар** существует связь "многие ко многим",



то есть каждая книга может быть положена в разные корзины, и в каждой корзине может быть много книг, при этом в каждой корзине учитывается количество одинаковых книг. Будем также хранить дату, когда книга была положена в корзину.

У каждого заказа имеется дата, способ доставки и владелец заказа. Таким образом, объект **покупатель** связан с заказом связью "один ко многим". С каждым заказом покупатель может согласиться на доставку бесплатного каталога – сохраним эту информацию в свойстве «бонус». При регистрации покупатель сообщает информацию о себе, а также может подписаться на одну из почтовых рассылок – предусмотрим для этого атрибут «рассылка».

Реляционная модель

Преобразуем ER-модель в реляционную модель. Получим следующие таблицы.

Товар(НомТов, НазТов,КолСтр, Автор, Цена, Обложка, НомИзд, НомКат) - столбец НомТов будет внешним ключом для таблиц СоставКорзины и СоставЗаказа.

Издатель(НомИзд, НазИзд) - столбец НомИзд является внешним ключом для таблицы Товар.

Категория(НомКат, НазКат) - столбец НомКат является внешним ключом для таблицы Товар.

СоставКорзины(НомКорз, НомТов, Колво, ДатаКорз) – отдельную таблицу Корзина не создаем. т.к., у объекта Корзина нет ничего, кроме идентификатора.

Заказ(НомЗак, ДатаЗак, Доставка, Бонус) столбец НомЗак является внешним ключом для таблицы СоставЗаказа.

СоставЗаказа(НомЗак, НомТов, Колво)

Покупатель(НомПок, Фамилия, Имя, Адрес, E-mail, Логин, Пароль, Рассылка)

Создание БД в MySQL

СУБД **MySQL** разработал **Майкл (Монти) Видениус**. Это небольшая и очень эффективная для своего размера реляционная СУБД, основанная на традициях **Hughes Technologies Mini SQL (mSQL)**.

Для работы с **MySQL** нужно выполнить следующее:

1. Разархивировать дистрибутив в произвольный каталог, запустить **Setup.exe** и установить **MySQL** в какой-нибудь каталог (например, **C:\MySQL**).
2. Если сервер установлен как служба, то он будет запущен автоматически. Если сервер установлен как приложение, то его следует запустить вручную. В любом случае полезно запустить программу **WinMysqlAdmin.exe** – простой интерфейс для администратора. На панели задач появится светофор: если он зеленый, то сервер запущен.
3. Запустить клиентскую часть **MySQL** - файл **C:\MySQL\BIN\mysql.exe**. Откроется консольное приложение, в котором можно набирать и выполнять команды **MySQL**. Для создания и заполнения базы данных нужно, как минимум:
    1. создать базу данных командой **CREATE DATABASE имя_БД;** (не забывайте в конце команд ставить точку с запятой!),
    2. выбрать текущую базу данных командой **USE имя_БД;** ,
    3. создать таблицы командой **CREATE TABLE ...;** ,
    4. заполнить таблицы данными командой **INSERT INTO ...;** ,

    Можно написать файл-скрипт, содержащий команды **MySQL** и выполнить его командой **source полное_имя_скрипта** (точку с запятой здесь ставить не надо!).

Более подробную информацию о составных частях **MySQL**, правах доступа, диалекте языка SQL, применяемом в данной СУБД, и т.п., можно

прочитать, например, в **Учебнике по mySQL**, расположенном по адресу **(http://kek.ksu.ru/EOS/PHP/mySQL/index.html).**

Сценарий создания и заполнения базы данных для нашего электронного магазина может выглядеть следующим образом.

```
create database books;
use books;

create table publishers
( id_publ int(5) primary key auto_increment,
  name_publ varchar(50));

insert into publishers (name_publ) values ("Питер");
insert into publishers (name_publ) values ("BHV");
insert into publishers (name_publ) values ("ЛОРИ");
insert into publishers (name_publ) values ("Диалектика");
insert into publishers (name_publ) values ("Que");

create table categories
( id_cat int(5) primary key auto_increment,
  name_cat varchar(50));

insert into categories (name_cat)
     values ("Компьютерная литература");
insert into categories (name_cat)
     values ("Художественная литература");
insert into categories (name_cat)
     values ("Справочники");
insert into categories (name_cat)
     values ("Иностранные языки");
insert into categories (name_cat)
     values ("Кулинария");

create table books
( id_book int(5) primary key auto_increment,
  name_book varchar(100),
  id_publ int(5),
  id_cat int(5),
  author varchar(50),
  pages int(4),
  price int(4),
  dat int(4),
  image varchar(50)
);

insert into books
(name_book,id_publ,id_cat,author,pages,price,dat,image) values
```

```
("Ajax в действии", 4, 1, "Дейв Крейн, Эрик Паскарелло, Даррен
Джеймс", 640, 200, 2006, "1216642298_145214.png");
insert into books
(name_book,id_publ,id_cat,author,pages,price,dat,image) values
("Microsoft Visual Studio 2008", 2, 1, "Пауэрс Л., Снелл М.",
1192, 400, 2009, "4a98c94faa176.jpg");

insert into books
(name_book,id_publ,id_cat,author,pages,price,dat,image) values
("Изучаем Ajax", 1, 1, "Бретт Маклафлин", 425, 300, 2008,
"1224319675_izuchaem-ajax.jpg");

insert into books
(name_book,id_publ,id_cat,author,pages,price,dat,image) values
("Гибкая разработка веб-приложений в среде Rails", 1, 1, "Д.
Томас, Д. Х. Хэнссон", 720, 400, 2008,
"1217485667_1000657293.jpg");

insert into books
(name_book,id_publ,id_cat,author,pages,price,dat,image) values
("Microsoft Visual C# в задачах и примерах", 2, 1, "Н. Культин",
314, 140, 2006, "4a3de5b9e8517.jpg");

create table basket_books
( id_bask char(15),
  id_book int(5),
  kolvo int(2),
  date_bask date);

create table customers
( id_cust int(5) primary key auto_increment,
  fam varchar(30),
  im varchar(30),
  addr varchar(100),
  mail varchar(30),
  login varchar(10),
  pass varchar(10),
  subscribe int(1));

create table orders
( id_order char(15),
  date_ord date,
  id_cust int(5),
  dostavka int(1),
  bonus int(5));

create table order_books
( id_order char(15),
  id_book int(5),
  kolvo int(2));
```

Основы языка серверных сценариев PHP

Как возник PHP

Годом рождения **PHP** можно считать 1995 год, когда независимый программист **Расмус Лердорф** написал скрипт для подсчета количества посетителей страницы со своим онлайновым резюме. Тот первый скрипт был написан на языке **Perl**, а так как средств для разработки Internet-приложений тогда было немного, то он вызвал живой интерес программистов и пользователей со всего мира.

Сначала **Лердорф** назвал свою программу **Personal Home Page**, а в дальнейшем, когда ее функции вышли далеко за пределы простого оформления домашней страницы, эту аббревиатуру стали расшифровывать рекурсивно, как **PHP Hypertext Processor**.

Вскоре появилась версия **PHP 2.0**, написанная на **C**. В разработке версии **3.0** уже участвовала целая команда авторов. Сейчас **PHP** - это не только язык серверных web-сценариев, но и творческое сообщество единомышленников, работающих над развитием своего программного продукта, который завоевал признание во всем мире. По сведениям **NETCRAFT (http://www.netcraft.com) PHP** используется на более чем 1 000 000 хостов.

Этот язык, вобравший в себя лучшие черты **Perl**, **C**, **Java**, создавался специально для web-приложений и получился очень удачным - лаконичным, простым, эффективным и надежным. Рассматриваемая в пособии версия **4.0** содержит такие возможности, как динамическое создание изображений, работу с **PDF**-файлами, поддержку многих форматов баз данных, интеграцию с **XML**, использование электронных платежных систем и т.п.

Существенным преимуществом языка **PHP** является тот факт, что в пределах одного web-документа можно перемешивать **HTML**-тэги и **PHP**-команды. Для того чтобы отделить **PHP**-код от **HTML**-тэгов, его ограничивают символами **<? ?>.**

Работает все это так. Сначала на сервере выполняются **PHP**-команды. Они могут содержать обращение к базам данных, работу с файлами и прочие виды обработки информации. Очень удобными являются команды **print** и **echo**, которые выполняют печать непосредственно в **HTML**-документ. После

того как документ полностью обработан на сервере, он отправляется пользователю и отображается у него в браузере. Таким образом, пользователь никаких серверных команд уже не увидит – он увидит только результаты их работы.

Что нужно для работы с PHP

PHP может работать с разными web-серверами: **Apache**, **Microsoft IIS(PWS)**, **Netscape Enterprize Server**, **Stronghold**, **Zeus** и, следовательно, на разных платформах: **Unix**, **Solaris**, **FreeBSD**, **Windows 9x/NT/XP**.

Рассмотрим установку **PHP** под конфигурацию **Windows+Apache** (может быть, экзотичную с точки зрения реального web-сайта, но очень удобную для использования на локальном компьютере, особенно для учебных целей).

Прежде всего, следует установить и сконфигурировать **Apache**. Установка **Apache**, как и любой другой программы под **Windows**, не вызывает особых проблем. Если в процессе установки не были созданы (это зависит от версии) ярлыки для запуска и останова программы в главном меню, можно создать их вручную. Для запуска используется файл **apache.exe**, для останова - тот же файл с параметрами **apache.exe -k shutdown**. По умолчанию **Apache** устанавливается в каталог **C:\Program Files\Apache Group\Apache.**

После установки следует открыть файл конфигурации **httpd.conf** (находится в подкаталоге **CONF**) и сделать в нем следующие исправления.

Строка

```
ServerAdmin yourname@yoursite.com
```

указывает почтовый адрес администратора сайта. Если сайт действующий, следует указать реальный адрес.

В качестве имени сервера для локального компьютера следует использовать **localhost**:

```
ServerName localhost
```

Теперь можно запустить **Apache** через главное меню или проводник. Затем запустите браузер и введите адрес **http://localhost**. Если установка

была выполнена успешно, то откроется страница с фирменным перышком **Apache**.

Теперь можно устанавливать **PHP**. Распакуйте архив **PHP** в произвольный каталог, например, **C:\PHP**.

Найдите среди распакованных файлов **php.ini-dist** (в версии 3.0 его имя **php3.ini-dist**). Это файл конфигурации **PHP**, более подробно он будет рассмотрен в главе "Конфигурация PHP" Скопируйте его в каталог **C:\WINDOWS** и переименуйте в **php.ini** (в версии 3.0 **php3.ini**).

Найдите файлы **php4ts.dll** и **Msvcrt.dll** и скопируйте их в каталог **C:\WINDOWS\SYSTEM**. Если файл **Msvcrt.dll** уже существует, перезаписывать его не надо.

Теперь нужно внести дополнительные изменения в файл **httpd.conf**. Найдите секцию, содержащую команды **ScriptAlias**. Здесь нужно создать строку

```
ScriptAlias /php4/ "C:/php/"
```

означающую псевдоним для корневого каталога **PHP**.

Найдите секцию **AddType**. Добавьте или раскомментируйте, если они уже существуют, строки

```
AddType application/x-httpd-php .phtml .php
AddType application/x-httpd-php-source .phps
```

Эти строки объявляют, что файлы с **PHP**-сценариями могут иметь расширения **phtml** и **php**.

Найдите секцию **Action**. Добавьте в нее строку

```
Action application/x-httpd-php /php4/php.exe
```

Эта строка определяет, какое именно приложение будет выполнять **PHP**-сценарии.

После этих изменений в файле конфигурации следует остановить **apache** и стартовать его заново.

Теперь можно протестировать **PHP**. Напишите файл с расширением **phtml**, содержащий следующий код (пример взят из главы "Переменные"):

```
<?
while (list($var, $value)=each($GLOBALS))
{
print "<br>$var => $value";
}
?>
```

(символами **<? ?>** код сценария отделяется от HTML-текста) и поместите его в каталог **HTDOCS** - корневой каталог **Apache**. Теперь запустите браузер и просмотрите этот файл через адрес **http://localhost**. Вы должны увидеть длинный список глобальных переменных и их значений.

Примечание. Специально для удобства разработчиков создан пакет Денвер, включающий в себя все необходимые нам программы – Apache, PHP и MySQL. Можете использовать этот пакет, но разбираться в тонкостях его работы будете сами!

Типы данных

В **PHP** используются следующие типы данных:

- целые числа
- вещественные числа
- строки
- логические величины
- массивы
- объекты

Целые числа

Так же, как и в языке **C**, для целых чисел кроме десятичного представления (**1, 16, 255**), допустимы восьмеричное (**01, 020, 0377**) и шестнадцатеричное (**0x1, 0x10, 0xff**).

Вещественные числа

Для чисел с десятичной точкой используется стандартная запись с фиксированной точкой (**1.5, 0.999**) и так называемая научная запись - с плавающей точкой (**0.15e1, 9.99e-1**).

Строки

Строки могут заключаться в двойные кавычки или в апострофы. Если в строке, заключенной в двойные кавычки, есть переменная, то вместо нее будет подставлено ее значение, например:

```
$year=2007;
$today="6 сентября $year года";
print $today;
```

Будет напечатано: **6 сентября 2007 года.**

В двойных кавычках также интерпретируются и, соответственно, подставляются управляющие символы, такие, как **\n, \t, \r** и т.п.

Логические величины

Нетрудно догадаться, что логический тип представлен значениями **true** и **false**.

Массивы

Обычные массивы, называемые в **PHP** *индексируемыми*, можно создавать с помощью инициализации отдельных элементов:

```
$month[0]="Январь";
$month[1]="Февраль";
$month[2]="Март";
```

При этом можно даже не указывать индексы, элементы массива создаются последовательно:

```
$month[]="Январь";
$month[]="Февраль";
$month[]="Март";
```

Для инициализации также удобно использовать функцию **array**:

```
$month=array("Январь","Февраль","Март");
```

Кроме индексируемых, в **PHP** используются *ассоциативные* массивы, в которых с каждым значением связывается строковый ключ:

```
$capital["Франция"]="Париж";
$capital["Великобритания"]="Лондон";
$capital["Япония"]="Токио";
```

При использовании функции **array** задаются пары ключ/значение:

```
$capital=array("Франция"=>"Париж",
               "Великобритания"=>"Лондон",
               "Япония"=>"Токио");
```

Массивы могут быть также многомерными.

Более подробно о функциях для работы с массивами рассказано в параграфе Массивы.

Объекты

Для определения класса используется команда **class** следующего вида:

```
class Light
{
var $isOn;

function click($OnOff)
{$this->isOn=$OnOff;}
}
```

Как видим, переменные-элементы класса нужно объявлять с помощью ключевого слова **var**, при обращении к ним следует использовать ключевое слово **this**. В целом определение класса очень напоминает язык **JavaScript**. Создать объект какого-либо класса можно только с помощью команды **new**.

```
$bulb = new Light;
```

Более подробно о работе с объектами рассказано в приложении Объекты.

Переменные

Переменные в **PHP** всегда начинаются со знака **$**. Имя переменной может содержать буквы, цифры и некоторые специальные символы и должно начинаться с буквы или знака подчеркивания. Регистр букв учитывается.

Переменные не обязательно объявлять. Любая переменная объявляется неявно при первом использовании. Тип переменной также неявно определяется по типу хранящегося в ней значения, т.е., в языке **PHP** переменные слабо типизированы. Явно переменную можно объявить с помощью ключевого слова **var**.

По умолчанию любая переменная является локальной. Для того чтобы изнутри функции обратиться к глобальной переменной, нужно объявить ее с префиксом **GLOBAL**:

```
GLOBAL $x;
```

или использовать системный ассоциативный массив **$GLOBAL**:

```
$GLOBALS["x"];
```

Статические переменные объявляются с ключевым словом **STATIC**:

```
STATIC $count=0;
```

Неявное преобразование (juggling) типа переменной происходит в том случае, если производится операция с переменными разных типов. Так, при выполнении арифметической операции над целым числом и строкой, представляющей собой целое число, результат будет целым числом. Если в операции участвуют целое и вещественное числа – результат будет вещественным числом, чтобы избежать потери точности.

Кроме неявного, можно использовать и **явное** преобразование типа (casting). Чтобы явно преобразовать выражение, нужно перед ним указать в скобках ключевое слово типа (точно так же, как в **C**):

- **int** или **integer;**
- **real, double** или **float;**
- **string;**
- **array;**
- **object.**

Если переменная преобразуется в массив, она становится первым (т.е., нулевым) элементом массива.

Если переменная преобразуется в объект, она становится атрибутом объекта и ей назначается имя **scalar**.

Переменную можно определить как ссылку на другую переменную:

```
$x=0;
$y=&$x;
```

При этом **$y** становится псевдонимом для **$x**, и при изменении **$y**, будет изменяться также и **$x**.

Массив **$GLOBALS**, который уже упоминался, содержит множество полезных стандартных переменных. Их имена и текущие значения можно распечатать, если последовательно просмотреть весь этот глобальный массив:

```
while (list($var, $value)=each($GLOBALS))
```

```
{
print "<br>$var => $value";
}
```

Знак **$** перед именем переменной на самом деле представляет собой операцию макроподстановки, т.е., вместо имени переменной подставляется ее значение. Например:

```
$first="something";
$second="first";
print $($second);
```

Что будет напечатано? Очевидно, строка **"something"**.

По окончании работы сценария все его переменные автоматически удаляются из памяти. Если же переменную нужно удалить раньше, можно использовать функцию **unset**:

```
unset ($MyArray);
```

Операции и операторы

В **PHP** используются следующие операции (смесь из языков **C** и **Perl**):

Обозначение	Ассоциативность	Описание
`()`	-	Изменение приоритета
`new`	-	Создание экземпляра объекта
`! NOT ~`	П	Логическое и поразрядное отрицания
`++ --`	П	Инкремент, декремент
`@`	П	Маскировка ошибок
`/ % *`	Л	Деление, остаток, умножение
`+ - .`	Л	Сложение, вычитание, сцепление строк
`<< >>`	Л	Поразрядный сдвиг влево и вправо
`< <= > =>`	-	Меньше, меньше или равно, больше, больше или равно
`== === != <>`	-	Равно, идентично, не равно

`&  ^  |`	Л	Поразрядные И, Исключающее ИЛИ, ИЛИ
`&&  ||`	Л	Логические И, ИЛИ
`?:`	П	Тернарная операция
`=  +=  -=  /=  %= *= .=  &=  |=  <<=  >>=`	П	Присваивание
`AND  XOR  OR`	Л	Логические И, Исключающее ИЛИ, ИЛИ
`//  #`	-	Однострочные комментарии
`/* */`	-	Многострочные комментарии

Операторы имеют 2 альтернативных синтаксиса:

Условный оператор	

`if (условие) {`
`блок`
`}`
`else {`
`блок`
`}`
	`if (условие) :`
	`блок`
	`else :`
	`блок`
	`endif;`

Цикл с предусловием	

`while (условие) {`
`блок`
`}`
	`while (условие) :`
	`блок`
	`endwhile;`

Цикл с постусловием	

`do {`
`блок`
`} while (условие);`
	`do :`
	`блок`
	`while (условие);`

Цикл со счетчиком	

`for (инициализация;`
`условие выхода;`
`действие по окончании итерации)`
`{`
`блок`
`}`
	`for (инициализация;`
	`условие выхода;`
	`действие по окончании итерации)`
	`:`
	`блок`
	`endfor;`

Цикл для перебора массивов (нумерованного и ассоциативного)	
`foreach ($массив as $элемент) {`	`foreach ($массив as $элемент) :`

блок }	блок endforeach;

foreach ($массив as $ключ => $элемент) {
блок
}
	foreach ($массив as $ключ => $элемент) :
	блок
	endforeach;

Оператор выбора	

switch (выражение) {
case (условие1):
блок
case (условие2):
блок
...
case (условиеN):
блок
default:
блок
}
	switch (выражение) :
	case (условие1):
	блок
	case (условие2):
	блок
	...
	case (условиеN):
	блок
	default:
	блок
	endswitch;

Оператор прерывания цикла	
break;	
Оператор прерывания итерации цикла	
continue;	
Операторы печати	
print строка; echo строка;	

Функции

PHP – это процедурный язык с небольшими объектно-ориентированными возможностями, поэтому в нем большое внимание уделяется процедурам, или функциям – в данном случае это одно и то же. Практически все в **PHP** реализовано через функции – от работы с массивами до интеграции с **XML** и электронными платежными системами (приятное разнообразие после тотальной объектно-ориентированности!). Большинство функций можно применять без специального объявления, но некоторые пакеты специфических функций требуют упоминания в секции **extensions**

файла **php.ini** (подробнее об этом см. главу Конфигурация PHP). Разумеется, программист может писать и собственные функции.

Формат функции в PHP следующий:

```
function имя_функции ([$параметр1, $параметр2, ..., $параметрN])
{
тело_функции
[return [возвращаемое_значение];]
}
```

Функции в **PHP** могут быть вложенными, т.е. одна функция может располагаться внутри другой (что недопустимо, например, в **C**). В этом случае внутреннюю функцию можно вызывать только после вызова внешней.

При вызове функций точно так же, как и при работе с переменными, можно применять макроподстановку **$**.

```
function english()
   { print "Good morning!"; }
function french()
   { print "Bon matin!"; }

$language="french";
$language(); // Вызов функции french
```

Массивы

Краткое описание массивов см. в параграфе "Типы данных".

Для работы с массивами разработано много полезных функций. Кратко рассмотрим самые часто используемые из них. Если тип параметра или возвращаемого значения функции не указан, значит, он может быть разный.

Создание и изменение массивов	
`array array ([значение1, значение2, ..., значениеN])`	создает массив из указанных значений
`void list (переменная1 [,переменная2, ... , переменнаяN])`	используется в **левой** части оператора присваивания (в правой должен быть массив), и присваивает своим аргументам-переменным значения элементов этого массива

array range (int мин_значение, int макс_значение)	создает массив из целых чисел заданного диапазона
int array_push (array массив, значение1 [, значение2,..., значениеN])	добавляет новые элементы в конец массива
void array_pop (array массив)	удаляет элемент с конца массива
void array_shift (array массив)	удаляет элемент с начала массива
void array_unshift (array массив, значение1 [, значение2,..., значениеN])	добавляет новые элементы в начало массива
array array_reverse (array массив)	возвращает перевернутый массив
array array_merge (array массив1, array массив2,... ,array массивN)	объединяет несколько массивов в один и возвращает его
array array_slice (array массив1, int смещение [, int длина])	возвращает "срез" массива, заданный смещением и длиной
Поиск в массиве	
bool in_array (элемент, array массив)	проверяет, содержится ли элемент в массиве
Просмотр элементов массива	
reset (array массив)	перемещает внутренний указатель массива на начало и возвращает его первый элемент
array each (array массив)	возвращает пару ключ/значение из текущего элемента массива и перемещает указатель к следующему элементу
void end (array массив)	перемещает внутренний указатель на конец массива
next (array массив)	перемещает внутренний указатель на один элемент вперед и возвращает значение этого элемента; возвращает ложь, если вышли за пределы массива
prev (array массив)	перемещает внутренний указатель на один элемент назад

	и возвращает значение этого элемента; возвращает ложь, если вышли за пределы массива
Размер массива	
int sizeof (array массив)	возвращает размер массива
int count (переменная)	если это массив, возвращает размер массива; если это скалярная переменная, возвращает 1; если переменная не существует, возвращает 0
array array_count_value (array массив)	подсчитывает число вхождений каждого значения и возвращает пары значение/количество в виде ассоциативного массива

Сортировки	
sort **rsort** **asort** **arsort** **ksort** **krsort** **usort** **uasort** **uksort**	всевозможные сортировки на все случаи жизни - по возрастанию и по убыванию, по значениям и по ключам
void shuffle (array массив)	перемешивает (на месте) значения элементов массива в случайном порядке

Строки

Строки являются самым распространенным типом данных в **PHP**. Для работы со строками написано очень много функций, рассмотрим наиболее полезные из них.

string trim (string строка)	удаляет лишние пробелы и символы \n, \t, \v, \b в начале и в конце строки
int strlen (string строка)	возвращает длину строки

int strcmp (string строка1, string строка2)	сравнивает строки и возвращает либо 0, либо положительное, либо отрицательное число (как в **C**)
int strcasecmp (string строка1, string строка2)	сравнивает строки и возвращает либо 0, либо положительное, либо отрицательное число (как в **C**), при этом регистр символов не учитывается
void parse_str (string строка)	выделяет в строке пары "переменная=значение" и присваивает данные значения реальным переменным. Удобно использовать при разборе строки параметров, полученной из URL.
array explode (string разделитель, string строка [, int количество])	разбивает строку, содержащую разделители, на подстроки и возвращает массив строк
string implode (string разделитель, array подстроки)	объединяет подстроки из массива в единую строку, помещая между подстроками разделители (хотя подстроки можно сцепить просто операцией конкатенации ".")
int strpos (string строка, string подстрока [, int смещение])	находит в строке первое вхождение подстроки и возвращает индекс первого символа подстроки (или **false**, если не найдено)
int strrpos (string строка, string символ)	находит в строке последнее вхождение данного символа и возвращает его индекс (или 0, если не найдено)
string str_replace (string подстрока, string новая_подстрока, string строка)	заменяет в строке все вхождения подстроки на новую подстроку
string substr (string строка, int начало [, int длина])	выделяет подстроку из строки
int substr_count (string строка, string подстрока)	подсчитывает количество вхождений подстроки в данную строку
string strtolower (string строка)	преобразует все буквы к нижнему регистру
string strtoupper (string строка)	преобразует все буквы к верхнему регистру
string ucfirst (string строка)	преобразует к верхнему регистру первую

	букву строки
string ucwords (string строка)	преобразует к верхнему регистру первую букву каждого слова

Также имеется набор функций для преобразования строк к **HTML**-формату и в обратную сторону.

Этап 3. Разработка шаблонов и главной страницы

Шаблоны

Шаблоном, применительно к web-программированию, можно назвать часть web-документа, которая используется в нескольких страницах. Шаблоны позволяют быстро проводить модификацию всего сайта – достаточно изменить информацию или оформление в шаблоне, и это отразится на всех страницах, использующих данный шаблон.

Для включения текста шаблона в документ используются функции **include** и **require.**

include (string имя_файла)	включает содержимое файла в сценарий.
include_once (string имя_файла)	включает содержимое файла в сценарий только один раз на протяжении сценария. Если файл уже был включен, то повторное включение игнорируется

Вызов этой функции можно поместить в блок **if-else**, например

```
<?
if ($user=="admin')
{
   include("adminMenu.phtml");
}
else {
   include("guestMenu.phtml ");
}
?>
```

Заметим, что фигурные скобки здесь обязательны, так как файл, включаемый в сценарий, как правило, состоит из нескольких строк.

require (string имя_файла)	включает содержимое файла в сценарий независимо от каких-либо условий. Даже если вызов функции находится в блоке if и условие ложно, файл все равно будет включен.
require_once (string имя_файла)	аналогично require, но включает содержимое файла в сценарий только один раз на протяжении сценария. Если файл уже был включен, то повторное включение игнорируется

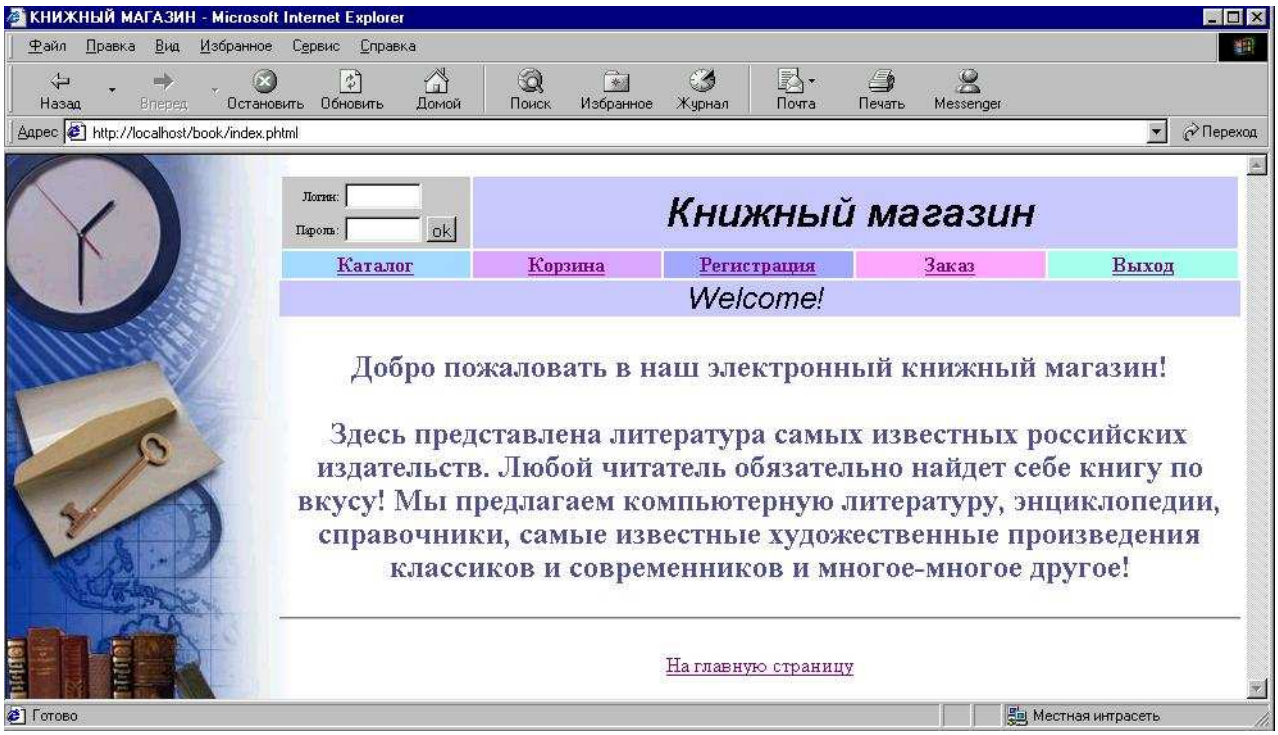
Обычно в отдельный шаблон удобно выделить шапку страницы, а в другой шаблон - нижний колонтитул или средства навигации.

```
<? include("shapka.phtml"); ?>
```

```
текст HTML-документа
```

```
<? include("bottomMenu.phtml"); ?>
```

Проанализируем, какие элементы оформления в нашем электронном магазине обязательно будут появляться на каждой странице.



К верхнему шаблону можно отнести фоновый рисунок, логотип вместе с маленькой формой для авторизации, а также строку меню. Обратите внимание, что заголовок текущего режима работы – "Welcome!", "Каталог", "Корзина" и.т.п., на полоске соответствующего цвета также можно вынести в шаблон. Для этого заведем PHP-переменные **`$color`** и **`$title`**, которые будем инициализировать в соответствующих страницах, а распечатывать – в верхнем шаблоне.

Итак, верхний шаблон **header.phtml** будет выглядеть следующим образом

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
     <title>КНИЖНЫЙ МАГАЗИН</title>
</head>

<body background="EULA.jpg" style="background-repeat:repeat-y"
leftmargin="130" rightmargin="5" bgProperties=fixed>
<table border="0" align="right" width="90%" cellpadding="0"
cellspacing="0">
<tr><td>

<table border="0" align="right" width="100%" >
<tr>
<td align="center" bgcolor="#cccccc">
<form action="auto.phtml">
<table>
<tr><td align="right"><font size=-2>Логин:</font></td>
<td align="left"><input type=text style="width:60; height:20;"
name=login></td></tr>
<tr><td align="right"><font size=-2>Пароль:</font></td>
<td align="left"><input type=password style="width:60;
height:20;" name=pass>
<input type=submit value=ok style="height:20;"></td></tr>
</table>
</td>
</form>
<td colspan="4" align="center" bgcolor="#ccccff">
<font face="Arial" size="+3"><i><b>Книжный
магазин</b></i></font></td></tr>
<tr><td align="center" bgcolor="#aaddff" width="20%">
<a href="catalog.phtml"><b>Каталог</b></a></td>
<td align="center" bgcolor="#ddaaff" width="20%">
<a href="basket.phtml"><b>Корзина</b></a></td>
<td align="center" bgcolor="#aaaaff" width="20%">
<a href="reg.phtml"><b>Регистрация</b></a></td>
<td align="center" bgcolor="#ffaaff" width="20%">
<a href="order.phtml"><b>Заказ</b></a></td>
<td align="center" bgcolor="#aaffee" width="20%">
<a href="exit.phtml"><b>Выход</b></a></td>
</tr>
</table>
</td></tr>
<tr><td align="center"
bgcolor=<?print $color?>><font face="Arial" size="+2">
<i><?print $title?></i></font><br>
</td></tr>
```

В нижний шаблон **footer.phtml** вынесем **гиперссылку "На главную страницу"** и закрывающие тэги документа.

```
<tr><td><center><hr><br>
<a href="index.phtml">На главную страницу</a></center></td></tr>
</table>
</body>
</html>
```

Главная страница

Таким образом, код главной страницы будет выглядеть довольно лаконично.

```
<?
$title="Welcome!";
$color="#ccccff";
include("header.phtml");
?>
<tr><td>
<center><h2><font color="#555599"><br>Добро пожаловать в наш
электронный книжный магазин! <br><br>Здесь представлена
литература самых известных российских издательств. Любой
читатель обязательно найдет себе книгу по вкусу! Мы предлагаем
компьютерную литературу, энциклопедии, справочники, самые
известные художественные произведения классиков и современников
и многое-многое другое!</font></h2></center>
</td></tr>

<? include("footer.phtml"); ?>
```

Обратите внимание, что для удобства размещения элементов оформления используется таблица, которая открывается в верхнем шаблоне, закрывается в нижнем шаблоне, а ее основная часть будет разной в разных режимах работы. Может быть, такая структура покажется вам довольно запутанной, но со временем вы сможете ее оценить, в особенности, если вам придется изменять общее оформление всего сайта.

Этап 4. Разработка витрины электронного магазина

PHP и MySQL

Для **MySQL** в **PHP** разработаны функции прямого доступа к данным. Рассмотрим функции, без которых нам не обойтись.

1. Подключение к серверу

int mysql_connect (string имя_хоста [:порт] [, string пользователь[, string пароль]])	функция пытается установить соединение с сервером, в случае успеха возвращает дескриптор соединения, в случае неудачи - ноль.

Например,

```
mysql_connect("localhost", "root", "") or
     die ("Не могу подключиться к серверу MySQL!");
```

В этой команде, если подключение не удалось, **mysql_connect** возвращает 0, и поэтому выполняется второй операнд логического ИЛИ - функция **die** выводит сообщение и завершает работу сценария.

2. Выбор базы данных

int mysql_select_db (string имя_БД [, int дескриптор_соединения])	функция пытается выбрать базу данных, в случае успеха возвращает истину, в случае неудачи - ложь. Если дескриптор соединения не указан, имеется в виду последнее соединение.

Например,

```
mysql_select_db("mycross") or
     die ("Не могу подключиться к базе данных");
```

В этой команде, если база данных не выбрана, **mysql_select_db** возвращает 0, и поэтому выполняется второй операнд логического ИЛИ - функция **die** выводит сообщение и завершает работу сценария.

3. Выполнение запроса

int mysql_query (string запрос	функция пытается выполнить запрос, в случае успеха возвращает дескриптор

[, int дескриптор_соединения])	результата-курсора, в случае неудачи - ноль. Если дескриптор соединения не указан, имеется в виду последнее соединение.

Например,

```
$result=mysql_query("SELECT kod, nazv FROM cross")
          or die("Не могу выполнить запрос!");
```

Если запрос неудачен, **mysql_query** возвращает 0, и поэтому выполняется второй операнд логического ИЛИ - функция **die** выводит сообщение и завершает работу сценария.

4. Получение очередной строки из результата запроса

array mysql_fetch_array (int дескриптор_результата)	функция пытается получить из результата запроса очередную строку и записывает значения полей в возвращаемый массив. Если строк больше нет, возвращается ноль.

Например,

```
while($row=mysql_fetch_array($result))
{
     print $row["kod"].",".$row["nazv"]."<br>";
}
```

Перебираем в цикле последовательно все строки курсора и печатаем их.

5. Закрытие соединения.

int mysql_close ([int дескриптор_соединения])	функция закрывает соединение с указанным дескриптором. Если дескриптор соединения не указан, имеется в виду последнее соединение.

Например,

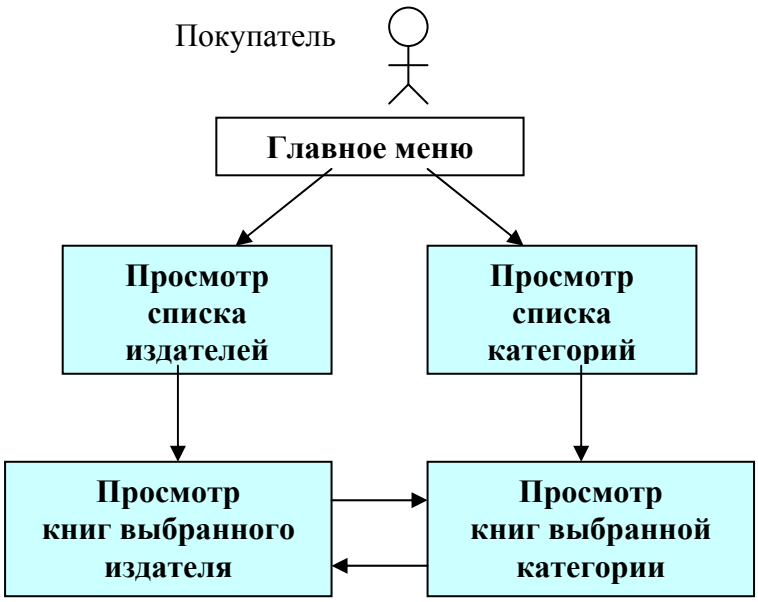
```
mysql_close();
```

Это самая простая последовательность действий, а вообще функций для работы с **MySQL** более 30 штук.

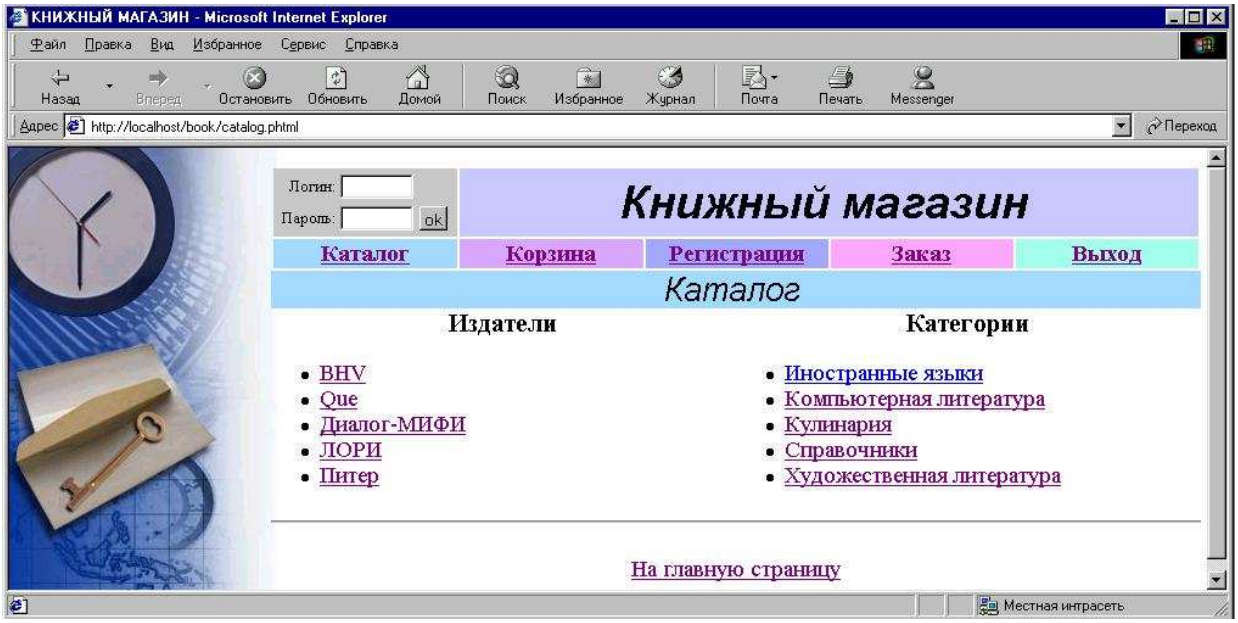
Просмотр издателей и категорий

Итак, приступим к разработке витрины нашего электронного магазина. Вспомним схему 2 со стр. 7 – действия с каталогом.

Схема 2. Работа с каталогом



Мы предполагали, что у читателя должна быть возможность выбора издателя и категории, поэтому самое удобное, что мы можем сделать – вывести список издательств и список категорий, как показано на следующей картинке.



Подключение к базе данных выглядит следующим образом (кстати, этот код удобно выделить в отдельный шаблон **connect.phtml**):

```
mysql_connect("localhost", "root", "") or
          die ("Не могу подключиться к серверу!");
mysql_select_db("books") or
          die ("Не могу подключиться к базе данных!");
```

Выполнение двух запросов выглядит так:

```
$strSQL1="SELECT * FROM publishers ORDER BY name_publ";
$result1=mysql_query($strSQL1) or
           die("Не могу выполнить запрос!");
$strSQL2="SELECT * FROM categories ORDER BY name_cat";
$result2=mysql_query($strSQL2) or
           die("Не могу выполнить запрос!");
```

Осталось вывести результаты запросов в две ячейки таблицы. В следующем листинге перемешаны **HTML**-тэги и **PHP**-команды.

```
<table border=0 width=100%>
<tr><td width="50%"><center><h3>Издатели</h3></center><ul>
<?
while($row=mysql_fetch_array($result1))
{?>
     <li><a href="show.phtml?type=1&id_publ=
     <?print $row["id_publ"];?>"><?print row["name_publ"];?></a>
<?}?>
</ul></td>

<td width="50%"><center><h3>Категории</h3></center><ul>
<?
while($row=mysql_fetch_array($result2))
{?>
     <li><a href="show.phtml?type=2&id_cat=
     <?print $row["id_cat"];?>"><?print $row["name_cat"];?></a>
<?}?>
</ul></td>
</tr>
</table>
```

Обратите внимание, как из названий издательств и категорий формируются гиперссылки, например:

```
<a href="show.phtml?type=1&id_publ=<?print $row["id_publ"];?>">
   <?print row["name_publ"];?></a>
```

Поскольку при щелчке по гиперссылке мы хотим получить список книг по заданному издателю или категории, то в сценарий **show.phtml** следует передать параметры **type** (если он равен 1, то выбран издатель, если 2 – то категория) и **id_publ** (код издателя) или **id_cat** (код категории).

Полный код сценария **catalog.phtml** приведен в Приложении 1.

Передача данных от браузера серверу

В последних абзацах предыдущего параграфа мы сформировали гиперссылки с параметрами, чтобы передать серверу информацию, какое издательство или какая категория книг нас интересуют. Возникает вопрос: как сервер принимает эти параметры?

Ответ на этот вопрос неоднозначен. Всё зависит от того, включен ли на сервере режим регистрации глобальных переменных. Если в файле **php.ini** установлен режим автоматической регистрации глобальных переменных

```
register_globals = on
```

то при вызове PHP-сценария по гиперссылке для каждого параметра создается серверная переменная с тем же именем, что у параметра. То есть, если мы создали гиперссылку

```
<a href="show.phtml?type=1&id_publ=2">BHV</a>
```

то в сценарии **show.phtml** будут существовать переменная с именем **$type** и значением 1, а также переменная с именем **$id_publ** и значением 2.

Тот же принцип применяется и для элементов формы (текстовых полей, списков, наборов радиокнопок, флажков), а также ключиков и сеансовых переменных (о двух последних понятиях вы узнаете из следующих параграфов). Для всех них также в этом случае автоматически создаются серверные переменные.

Но автоматическая регистрация серверных переменных считается небезопасной!

Поэтому рекомендуется устанавливать параметр

```
register_globals = off
```

При этом для обращения к переменным придется использовать разнообразные ассоциативные массивы, которые автоматически создаются на сервере:

$HTTP_POST_VARS["имя_элемента_формы"]	Массив содержит переменные, переданные методом POST
$HTTP_GET_VARS["имя_параметра_адресной_строки"]	Массив содержит переменные, переданные методом GET
$HTTP_COOKIE_VARS["имя_ключика"]	Массив содержит ключики
$HTTP_SESSION_VARS["имя_сеансовой переменной"]	Массив содержит сеансовые переменные

Именно такой подход мы и будем использовать в целях безопасности.

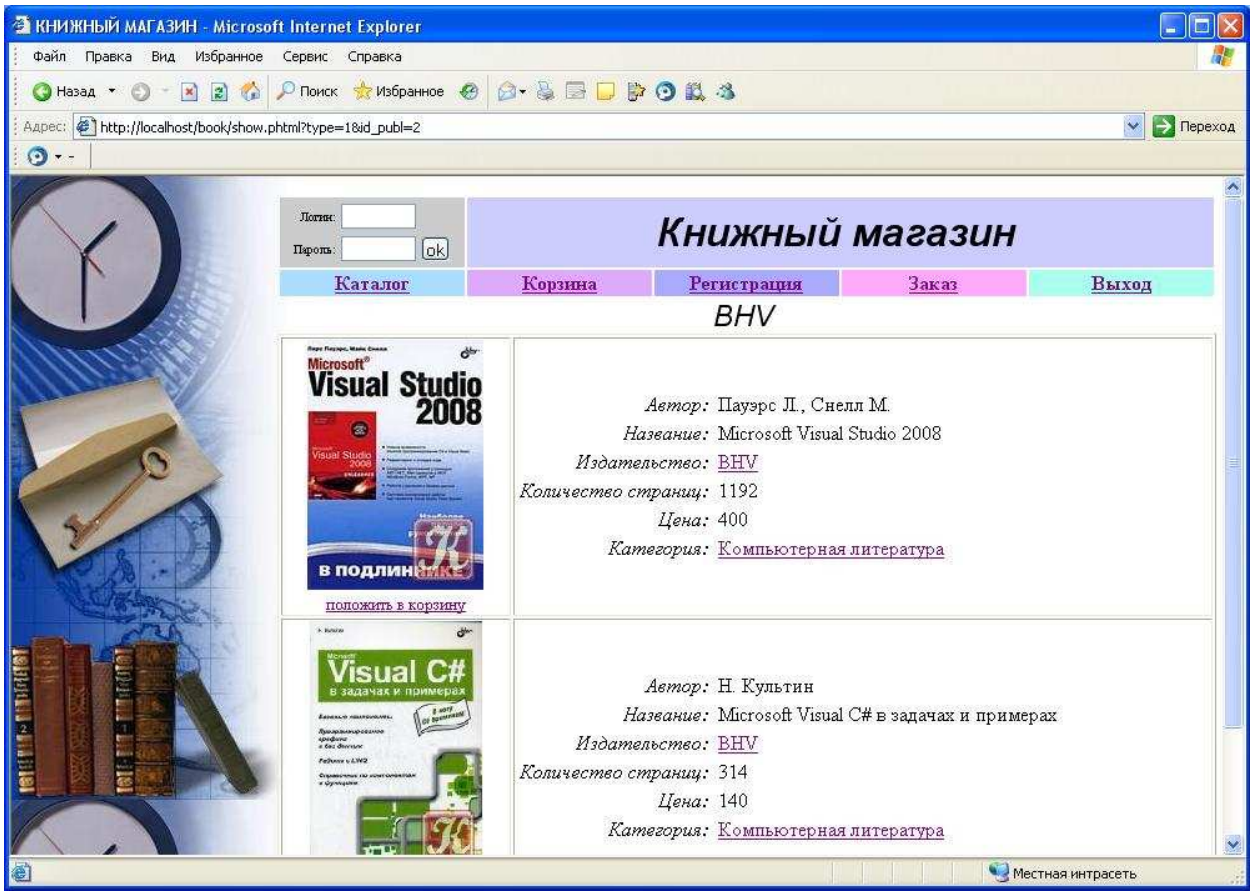
Примечание. Заметим, что если мы обратимся к несуществующей переменной (то есть к такой переменной, которая не была передана через форму или адресную строку, не является ключиком или сеансовой переменной и не была инициализирована в сценарии), то ошибки не возникнет! Просто эта переменная равна пустой строке (ее также можно сравнивать с нулем, результат такой операции будет истинным).

Просмотр списка книг

Сценарий **show.phtml** предназначен для просмотра списка книг. Прежде всего, прочитаем параметры, переданные из браузера:

```
$id_publ=$HTTP_GET_VARS["id_publ"];
$id_cat=$HTTP_GET_VARS["id_cat"];
$type=$HTTP_GET_VARS["type"];
```

Если мы хотим выбрать книги определенного издательства, мы передаем параметр **type=1,** и обращение к базе данных выглядит следующим образом:




```
if ($type==1)
{
$strSQL1="SELECT name_publ FROM  publishers
     WHERE id_publ=".$id_publ;
$result=mysql_query($strSQL1) or
        die("Не могу выполнить запрос1!");
if($row=mysql_fetch_array($result))
          $title=$row["name_publ"];
$strSQL1="SELECT id_book, image, author, name_book,
  books.id_publ, name_publ, pages, price, books.id_cat, name_cat
  FROM books, publishers, categories WHERE
  books.id_cat=categories.id_cat AND
  books.id_publ=publishers.id_publ AND books.id_publ=".$id_publ;
}
```

Если же мы хотим выбрать книги определенной категории, мы передаем параметр **type=2**, и обращение к базе данных выглядит следующим образом:

```
if ($type==2)
{
$strSQL1="SELECT name_cat FROM  categories
   WHERE id_cat=".$id_cat;
$result=mysql_query($strSQL1) or
        die("Не могу выполнить запрос1!");
if($row=mysql_fetch_array($result))
```

```
          $title=$row["name_cat"];
$strSQL1="SELECT id_book, image, author, name_book,
  books.id_publ, name_publ, pages, price, books.id_cat, name_cat
  FROM books, publishers, categories WHERE
  books.id_cat=categories.id_cat AND
  books.id_publ=publishers.id_publ AND books.id_cat=".$id_cat;
}
$result1=mysql_query($strSQL1) or
         die("Не могу выполнить запрос2!");
```

Затем формируем **HTML**-таблицу с информацией о книгах. Обратите внимание, как формируются тэги `<img>` для изображения обложек книг:

```
<img src="images/<?print $row["image"];?>"
     alt="<?print $row["name_book"];?>" border="0">
```

Под обложкой книги расположена гиперссылка **"положить в корзину"**. Эта гиперссылка вызывает сценарий **dobasket.phtml**, в котором объединены несколько режимов работы с корзиной. Для добавления книги в корзину мы передаем параметры **type=1** и **id_book** (код книги).

```
<a href="dobasket.phtml?type=1&id_book=
     <?print $row["id_book"];?>">
     <font size=-1>положить в корзину</font>
</a>
```

Кроме того, обратите внимание, что название издательства и категории представляют собой гиперссылки, с помощью которых можно быстро переключиться на другую классификацию.

```
<table border="1" width="100%" align="right" >
<?
while($row=mysql_fetch_array($result1))
{?>
  <tr>
  <td align="center"><img src="images/<?print $row["image"];?>"
   alt="<?print $row["name_book"];?>" border="0">
  <center><a href="dobasket.phtml?type=1&id_book=
     <?print $row["id_book"];?>"><font size=-1>
         положить в корзину</font></a></center></td>
  <td>
  <table>
  <tr><td align="right"><i>Автор: </i></td>
   <td><?print $row["author"];?></td></tr>
```

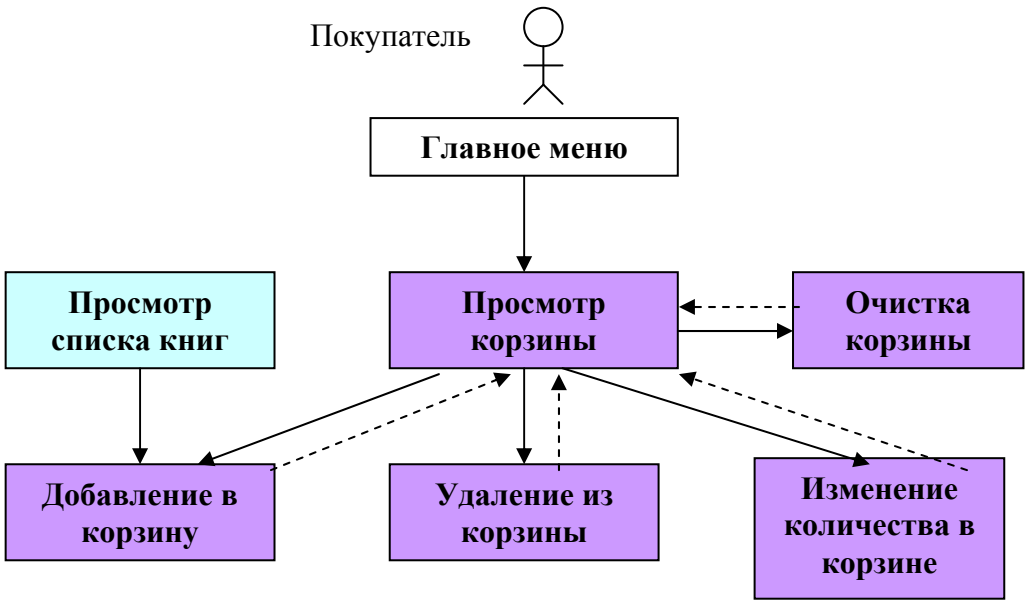
```
  <tr><td align="right"><i>Название: </i></td>
   <td><?print $row["name_book"];?></td></tr>
  <tr><td align="right"><i>Издательство: </i></td>
   <td><a href="show.phtml?type=1&id_publ=
         <?print $row["id_publ"];?>">
   <?print $row["name_publ"];?></a></td></tr>
  <tr><td align="right"><i>Количество страниц: </i></td>
   <td><?print $row["pages"];?></td></tr>
  <tr><td align="right"><i>Цена: </i></td>
   <td><?print $row["price"];?></td></tr>
  <tr><td align="right"><i>Категория: </i></td>
   <td><a href="show.phtml?type=2&id_cat=
         <?print $row["id_cat"];?>">
   <?print $row["name_cat"];?></a></td></tr>
  </table>

</td></tr>
<?}?>
</table>
```

Полностью код сценария приведен в Приложении 1.

Этап 5. Разработка корзины покупателя

Вспомним схему 3 со стр. 8 – действия с корзиной.

Схема 3. Работа с корзиной



При просмотре каталога покупатели непременно захотят отложить понравившиеся книги в корзину. При этом предварительной регистрации мы не требуем. Как же отличить корзину одного покупателя от корзины другого?

В таких случаях обычно используется стандартный подход анонимных корзин, предусматривающий хранение идентификатора корзины в ключике (cookie) на компьютере пользователя.

Cookies

Cookies, или, как их называют по-русски, **ключики**, представляют собой именованные кусочки информации, которые web-сервер может сохранить на клиентском компьютере. Рассмотрим коротко процесс передачи ключиков между сервером и клиентом. Он состоит из следующих шагов:

1. Сервер передает информацию о ключике, который он хочет создать: имя, значение, срок действия, имя домена (ключик всегда создается для конкретного домена) в служебной части **HTML**-страницы, в виде строки заголовка (header).

2. Браузер клиента записывает ключик в специальный файл на компьютере клиента. Ключик будет храниться там, пока не истечет срок его действия (для хранимых ключиков), либо пока браузер не будет закрыт (для временных ключиков).

3. При формировании запроса к web-серверу о получении какой-либо страницы браузер отправляет в заголовке запроса все ключики этого домена, которые хранятся на клиентском компьютере.

4. Сервер вместе с запросом страницы получает информацию о ключиках и может ее использовать по собственному усмотрению.

Ключики являются единственным средством сохранения информации из web-страницы на компьютере клиента. Необходимость в сохранении информации возникает, например, в том случае, если нужно отслеживать сеанс пользователя и однозначно его идентифицировать. Проблема здесь состоит в том, что отдельные **HTML**-страницы сайта очень слабо связаны друг с другом и для передачи информации между ними существует не так уж много способов.

Вернемся к электронному магазину. Пользователь, перемещаясь по страницам магазина, выбирает товары и кладет их в корзину. Для сохранения этой информации можно использовать следующие подходы:

- Использовать фреймовую структуру и хранить информацию в глобальных переменных на уровне **frameset** с помощью клиентских сценариев **JavaScript** или **VBScript**. Эти переменные будут доступны из любого фрейма, но, разумеется, будут уничтожены при закрытии окна браузера и даже просто при переходе на другой сайт *(по поводу фреймов и языка клиентских сценариев **JavaScript** – см. подробнее в электронном учебнике по Web-программированию* (http://kek.ksu.ru/EOS/TESTS/index.html)).

- Хранить информацию о корзине на сервере в базе данных. Этот способ позволяет сохранять информацию в течение любого срока. Для однозначной идентификации корзины пользователя нужно только присвоить ей уникальный номер. Этот номер можно передавать между

HTML-страницами в виде скрытого поля, а можно записать его в **cookies**. Именно этот подход мы и будем здесь использовать.

Создание ключика для идентификатора корзины

Итак, казалось бы, ключик для корзины разумно создавать при входе на главную страницу. Но пользователь может зайти на любую страницу нашего сайта, просто набрав ее адрес в адресной строке! Поэтому любая страница нашего сайта должна содержать следующие действия: следует либо создавать ключик, если он не существует, либо продлевать срок хранения ключика на заданный интервал, начиная с текущего момента. Установим, что срок хранения корзины равен двум неделям. Соответствующий сценарий удобно поместить в верхний шаблон **header.phtml,** тогда он наверняка будет выполняться при загрузке любой страницы сайта.

```
$id_bask=$HTTP_COOKIE_VARS["id_bask"];

if (!isset($id_bask))
{
     $uniq_ID=uniqid("ID");
     setcookie("id_bask", $uniq_ID, time()+60*60*24*14);
     // создадим ключик
}
else
     setcookie("id_bask", $id_bask, time()+60*60*24*14);
     // пересоздадим ключик с тем же значением, т. е.
     // продлим его срок хранения еще на 2 недели
```

Рассмотрим более подробно этот сценарий. Прежде всего, мы пытаемся прочитать ключик из глобального серверного массива и проверяем, не является ли он пустым. Для этого используется функция **isset**:

```
if (!isset($id_bask))
{ ... }
```

`int isset (mixed имя_ключика)`	проверяет, существует ли переменная с заданным именем. Если не существует, возвращает 0.

Если ключик не существует, нужно сгенерировать для него значение, а затем сохранить его на компьютере клиента. Для генерации значения удобно использовать функцию **uniqid**.

```
$uniq_ID=uniqid("ID");
```

int uniqid (string префикс [, boolean lcg])	создает уникальное значение с заданным префиксом. Если включен второй параметр, значение будет "более уникальным".

а затем собственно создать ключик:

```
setcookie("id_bask", $uniq_ID, time()+60*60*24*14);
```

int setcookie (string имя [, string значение [, int срок [, string путь [, sting домен [, int безопасность]]]]])	создает ключик с заданным именем.

Рассмотрим более подробно параметры этой функции.

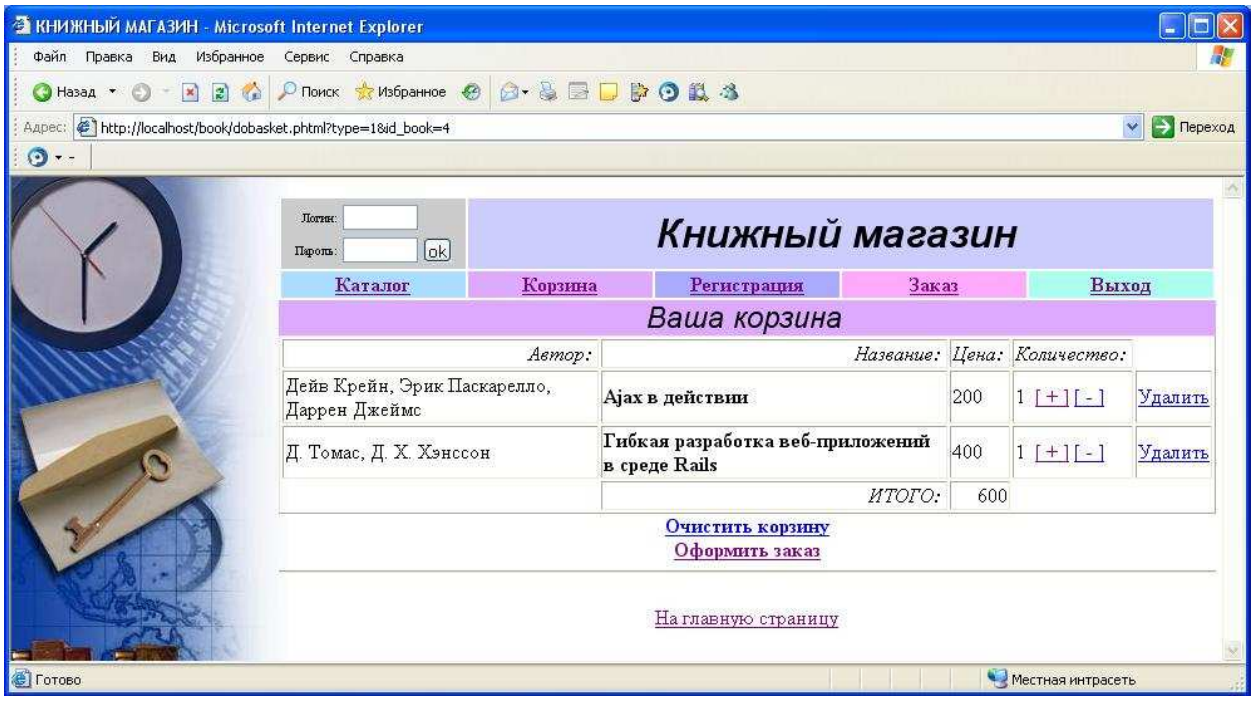
- **Имя** является обязательным параметром и подчиняется общим правилам имен для идентификаторов.
- **Значение** может быть любым.
- **Срок действия** измеряется в секундах; если срок не задан, то ключик является временным и уничтожается по завершении сеанса работы браузера.
- **Домен**, который создал данный ключик, может читать его значение. Для остальных доменов это запрещено. По умолчанию используется имя домена, от которого было получено значение этого ключика.
- **Безопасность** имеет значение истина/ложь и указывает, можно ли читать ключик в небезопасной среде.

Обратите внимание, что ключик создается до начала формирования текста страницы, т.е. до тега **<html>**. Более того, значение **cookie** должно устанавливаться **до** передачи в браузер **любой** другой информации, относящейся к странице (даже лишняя пустая строка или пробел перед началом серверного сценария приведет к ошибке!).

Для искусственного удаления ключика в качестве срока действия нужно указать уже прошедший момент времени (значение ключика можно задавать любое):

```
setcookie("id_bask", "", time()-600);
```

Просмотр корзины



В сценарии просмотра корзины **basket.phtml** прежде всего читаем значение ключика из глобального массива:

```
$id_bask=$HTTP_COOKIE_VARS["id_bask"];
```

Затем проверяем, не является ли корзина пустой:

```
<?
$strSQL1="SELECT COUNT(*) as count FROM basket_books WHERE
id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1) or die("Не могу выполнить
запрос1!");
$row=mysql_fetch_array($result1);
if($row["count"]==0)
{
?>
  <tr><td bgcolor='#ff9999' align='center'>
        <b>Ваша корзина пуста!</b></td></tr>
<?
}?>
```

Если корзина не пуста, выдаем ее содержимое на экран. Обратите внимание, как подсчитывается итоговая сумма. Обратите также внимание, как создаются гиперссылки «+» и «-».

```
<?
else
{
$strSQL1="SELECT image, author, name_book, pages, price, kolvo,
   id_bask, books.id_book FROM books, basket_books
      WHERE books.id_book=basket_books.id_book
      AND id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1) or die("Не могу выполнить
запрос2!");
?>
<tr><td>
<table border="1" width="100%" align="right" >
<tr><td align="right"><i>Автор: </i></td>
<td align="right"><i>Название: </i></td>
<td align="right"><i>Цена: </i></td>
<td align="right"><i>Количество: </i></td>
<td></td></tr>
<?
$sum=0;
while($row=mysql_fetch_array($result1))
{
?>
  <tr>
  <td><?print $row["author"];?></td>
  <td><b><?print $row["name_book"];?></b></td>
  <td><?print $row["price"];?></td>
  <td><?print $row["kolvo"];?>
  <a href="dobasket.phtml?type=1&id_book=
     <?print $row["id_book"];?>" title="Увеличить">[ + ]</a>
  <a href="dobasket.phtml?type=2&id_book=
     <?print $row["id_book"];?>" title="Уменьшить">[ - ]</a>
  </td>
  <td> <a href="dobasket.phtml?type=3&id_book=
     <?print $row["id_book"];?>">Удалить</a></td>
</tr>
<?
  $sum=$sum+$row["price"]*$row["kolvo"];
}?>
<tr><td align="right"></td><td align="right"><i>ИТОГО:
</i></td><td align="right"><?print $sum;?></td><td
align="right"></td></tr>
</table>
<?
}?>
```

Добавление товара в корзину

Когда в каталоге книг пользователь выбирает понравившуюся книгу и щелкает по ссылке **"положить в корзину"**, вызывается сценарий **dobasket.phtml**, в который передаются параметры **type=1** и **id_book** (код выбранной книги). Кроме того, в этом сценарии нам потребуется ключик **$id_bask,** который означает идентификатор корзины. Читаем эти данные из глобальных массивов:

```
$type=$HTTP_GET_VARS["type"];
$id_book=$HTTP_GET_VARS["id_book"];
$id_bask=$HTTP_COOKIE_VARS["id_bask"];
```

Алгоритм при добавлении товара в корзину выглядит следующим образом. Если эта книга уже присутствует в корзине, то мы только увеличиваем количество на единицу. В противном случае в корзину добавляем новую строку (*примечание:* функция **MySQL CURDATE()** возвращает текущую дату).

Обратите внимание, что при динамическом формировании запросов значения строковых констант (которые используются, например, в опции **WHERE**) следует помещать в одинарные кавычки.

```
if($type==1) // положить в корзину
{
     $strSQL="SELECT * FROM basket_books
       WHERE id_book=".$id_book." AND id_bask='".$id_bask."'";
     $result=mysql_query($strSQL) or
         die("Не могу выполнить запрос1!");
     if ($row=mysql_fetch_array($result))
     {
       $strSQL="UPDATE basket_books SET kolvo=kolvo+1
         WHERE id_book=".$id_book." AND id_bask='".$id_bask."'";
     }
     else
     {
       $strSQL="INSERT INTO basket_books (id_bask, id_book,
            Kolvo, date_bask) VALUES
            ('".$id_bask."',".$id_book.",1,CURDATE())";
     }
     mysql_query($strSQL);
}
```

После выполнения действий с базой данных мы хотим выдать пользователю на экран состав его корзины. Соответствующий сценарий просмотра корзины у нас уже есть. Поэтому последняя команда в сценарии **dobasket.phtml** заключается в подключении страницы **basket.phtml:**

```
<? include("basket.phtml"); ?>
```

Уменьшение и увеличение количества

Как видим при просмотре корзины, рядом с каждым товаром есть две гиперссылки – со значками "плюс" и "минус". Эти гиперссылки означают "увеличить количество товара в корзине на одну штуку" и "уменьшить количество товара в корзине на одну штуку".

При щелчке по гиперссылке "+" вызывается сценарий **dobasket.phtml** с параметром **type=1**, соответствующий код сценария мы уже рассмотрели при добавлении товара в корзину.

При щелчке по гиперссылке "–" вызывается сценарий **dobasket.phtml** с параметром **type=2**. Если количество было равно 1, то мы полностью удаляем данную книгу из корзины. Если же количество было больше 1, то мы просто переписываем строку в корзине, уменьшая количество на единицу.

```
if($type==2) // уменьшить количество
{
     $strSQL="SELECT * FROM basket_books WHERE
           id_book=".$id_book." AND id_bask='".$id_bask."'";
     $result=mysql_query($strSQL)
               or die("Не могу выполнить запрос1!");
     if ($row=mysql_fetch_array($result))
     {
          if ($row["kolvo"]>1)
          {
          $strSQL="UPDATE basket_books SET kolvo=kolvo-1 WHERE
             id_book=".$id_book." AND id_bask='".$id_bask."'";
          }
          else
          {
          $strSQL="DELETE FROM basket_books WHERE
             id_book=".$id_book." AND id_bask='".$id_bask."'";
          }
     }
     mysql_query($strSQL);
```

```
}
```

Удаление товара из корзины

При просмотре корзины также можно удалить любой товар из нее. Для этого служит гиперссылка **"Удалить"**, по которой вызывается тот же сценарий **dobasket.phtml**, но уже с параметром **type=3**:

```
if($type==3) // удалить из корзины
{
     $strSQL="DELETE FROM basket_books WHERE
         id_book=".$id_book." AND id_bask='".$id_bask."'";
     mysql_query($strSQL);
}
```

Кроме того, корзину можно полностью очистить, щелкнув по гиперссылке **"Очистить корзину"**. В этом случае опять вызывается тот же сценарий **dobasket.phtml**, но с параметром **type=4**:

```
if($type==4) // очистить корзину
{
     $strSQL="DELETE FROM basket_books WHERE
           id_bask='".$id_bask."'";
     mysql_query($strSQL);
}
```

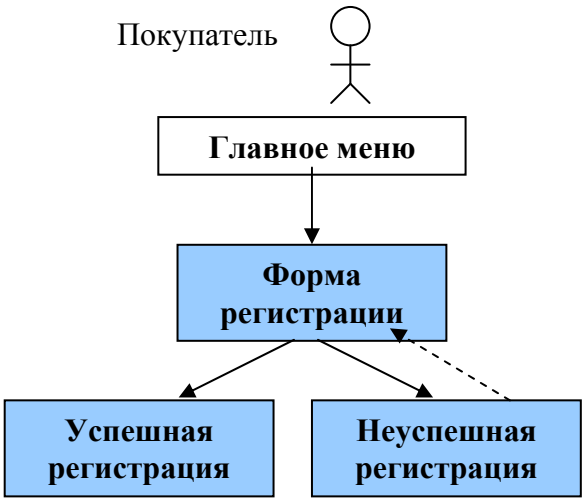
Полностью код сценария приведен в Приложении 1.

Этап 6. Разработка системы регистрации и авторизации посетителей

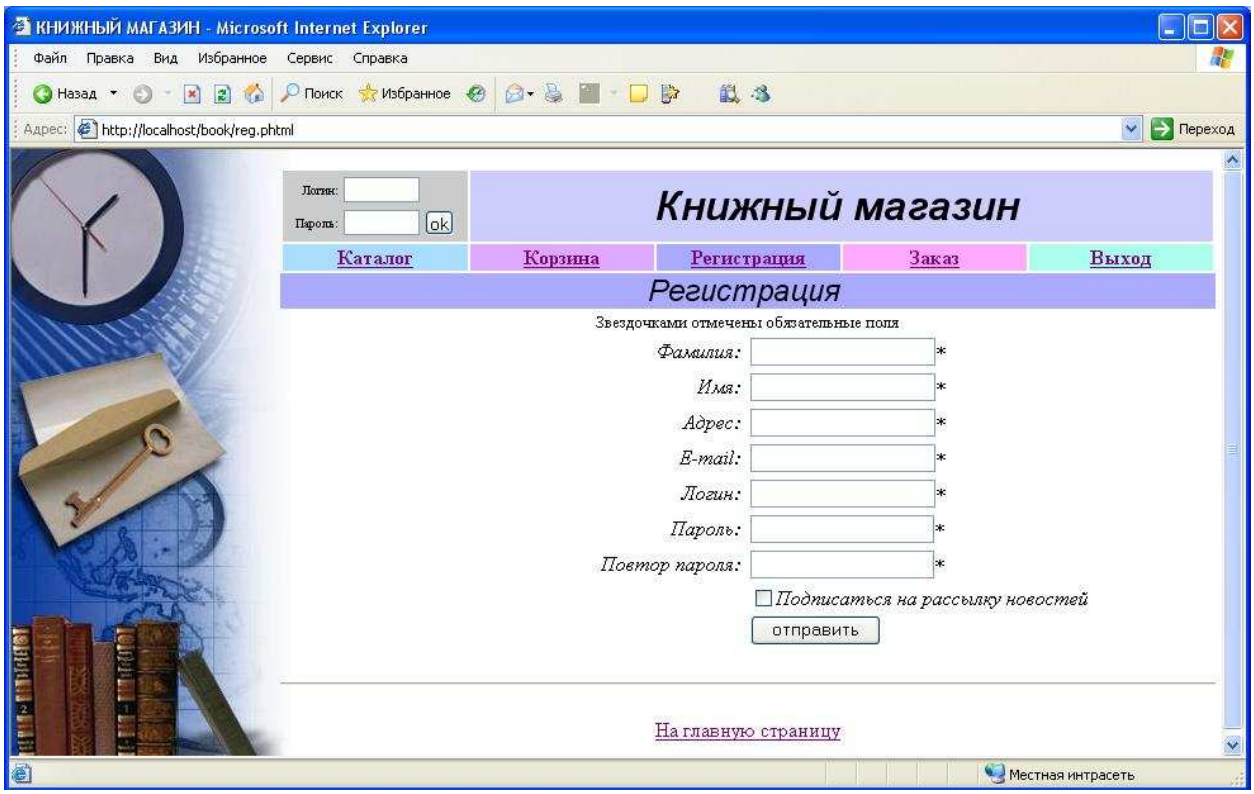
Регистрация

Вспомним схему 5 со стр. 9 – регистрацию покупателя.

Схема 5. Регистрация



Для регистрации будем использовать следующую форму (**reg.phtml**).



После того как посетитель нашего магазина отложил понравившиеся ему книги в корзину, возможно, он захочет их действительно купить. Для этого пользователь должен сообщить информацию о себе, т.е. зарегистрироваться. Обратите внимание, что кроме текстовых полей в форме есть флажок:

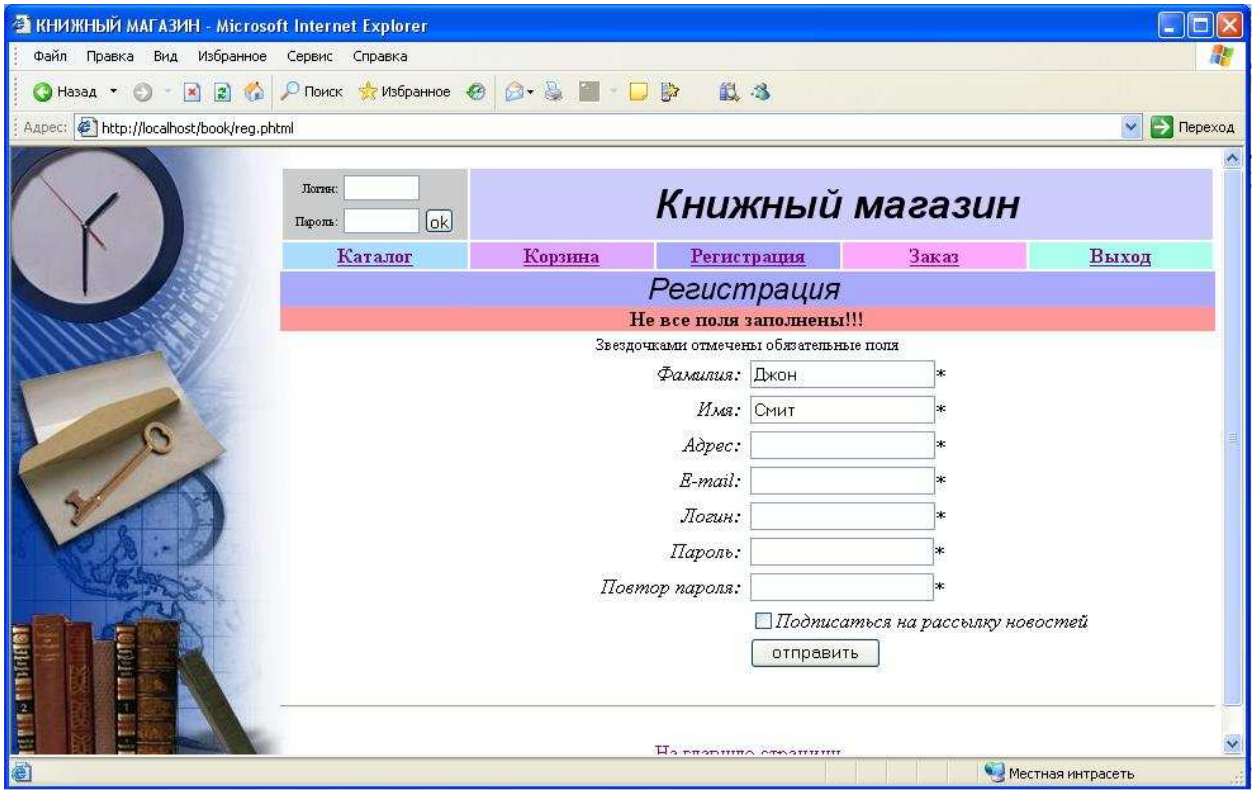
```
<input type="checkbox" value="1" name="subscribe">
<i>Подписаться на рассылку новостей</i>
```

При щелчке на кнопке "отправить" будет вызываться тот же сценарий **reg.phtml,** в который передаются все значения полей этой формы, включая скрытое поле с именем **type** и значением **1** (это скрытое поле нужно для того, чтобы при первой загрузке формы регистрации не выполнялось никаких проверок). В сценарии проверяется корректность заполнения полей формы:

```
$fam=$HTTP_POST_VARS["fam"];
$im=$HTTP_POST_VARS["im"];
$addr=$HTTP_POST_VARS["addr"];
$mail=$HTTP_POST_VARS["mail"];
$pass=$HTTP_POST_VARS["pass"];
$pass2=$HTTP_POST_VARS["pass2"];
$login=$HTTP_POST_VARS["login"];
$type=$HTTP_POST_VARS["type"];
$subscribe=$HTTP_POST_VARS["subscribe"];

// была нажата кнопка "отправить" ?
if($type==1)
{
// все поля не пустые ?
if($fam!="" && $im!="" && $addr!="" && $mail!="" && $login!=""
&& $pass!="" && $pass2!="")
{
   // поля пароля и повтора пароля не совпадают ?
   if($pass!=$pass2)
   {
    $message="<tr><td bgcolor='#ff9999' align='center'><b>
    Поля пароля и повтора пароля не совпадают!!!</b></td></tr>";
   }
   else
   {
     // ищем, нет ли в базе данных пользователя с таким логином
     $strSQL1="SELECT id_cust FROM customers WHERE
           login='".$login."'";
     $result1=mysql_query($strSQL1)
           or die("Не могу выполнить запрос!");
     // такой логин уже есть ?
```

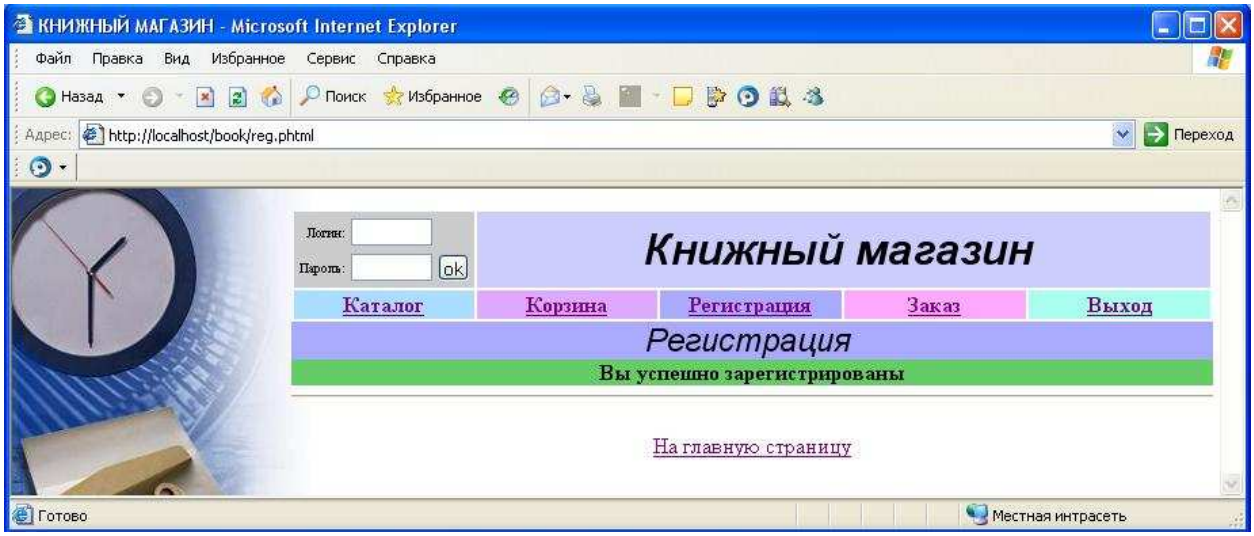
```
      if($row=mysql_fetch_array($result1))
      {
        $message="<tr><td bgcolor='#ff9999' align='center'>
        <b>Такой логин уже существует!!! Выберите другой
            логин</b></td></tr>";
      }
      else
      {
          // создаем нового пользователя
           $strSQL1="INSERT INTO customers
            (fam, im, addr, mail, login, pass, subscribe)
           VALUES('".$fam."','".$im."','".$addr."','".
           $mail."','".$login."','".$pass."','".$subscribe."')";
           $result1=mysql_query($strSQL1)
               or die("Не могу выполнить запрос!");
           $message="<tr><td bgcolor='#66cc66' align='center'>
           <b>Вы успешно зарегистрированы</b></td></tr>";
           $success=true;
           }
      }
}
else
$message="<tr><td bgcolor='#ff9999' align='center'><b>Не все
поля заполнены!!!</b></td></tr>";
}
```

В том случае, если пользователь заполнил форму с какими-либо ошибками, она снова выводится на экран для корректировки этих ошибок:



```
<form action=reg.phtml method=post>
<tr><td>
<table border="0" width="100%" align="right" >
<tr><td align="right"><i>Фамилия: </i></td><td>
  <input type=text name=fam value="<?print $fam?>"></td></tr>
<tr><td align="right"><i>Имя: </i></td><td>
  <input type=text name=im value="<?print $im?>"></td></tr>
<tr><td align="right"><i>Адрес: </i></td><td>
  <input type=text name=addr value="<?print $addr?>"></td></tr>
<tr><td align="right"><i>E-mail: </i></td><td>
  <input type=text name=mail value="<?print $mail?>"></td></tr>
<tr><td align="right"><i>Логин: </i></td><td>
 <input type=text name=login value="<?print $login?>"></td></tr>
<tr><td align="right"><i>Пароль: </i></td><td>
 <input type=password name=pass value=""></td></tr>
<tr><td align="right"><i>Повтор пароля: </i></td><td>
 <input type=password name=pass2 value=""></td></tr>
<tr><td></td>
<td><input type="checkbox" value="1" name="subscribe">
 <i>Подписаться на рассылку новостей</i></td></tr>
 <input type=hidden value=1 name=type>
<tr><td align="right"></td><td>
<input type=submit value="отправить"></td></tr>
</table>
</form>
```

Наконец, после правильного заполнения всех обязательных полей формы пользователь будет зарегистрирован:



Полностью код сценария приведен в Приложении 1.

Сессии

В качестве более безопасной альтернативы временным ключикам можно использовать такое понятие, как сессии. Сессия, или сеанс данных позволяет хранить практически неограниченное количество информации в сеансовых переменных. Технологически сессии тоже опираются на ключики; точнее говоря, один ключик – номер сессии – обычно хранится на клиенте, а вся остальная сеансовая информация – на сервере. Для работы необходимы, по крайней мере, следующие функции:

`bool` **`session_start`**`(void);`	стартует сессию. Всегда возвращает истину.
`bool` **`session_register`** `(mixed name [, mixed ...])`	регистрирует сеансовую переменную с именем **name**. Возвращает истину в случае успеха и ложь в случае неудачи. Эта переменная теперь доступна до конца сеанса или пока ее не удалят.
`bool` **`session_unregister`** `(string name)`	уничтожает сеансовую переменную с именем **name**.
`bool` **`session_destroy`**`(void);`	завершает сессию. Вся сеансовая информация уничтожается.

Также необходимо перед началом работы в файле **php.ini** установить параметр для автостарта сессии:

```
session.auto_start = 1
```

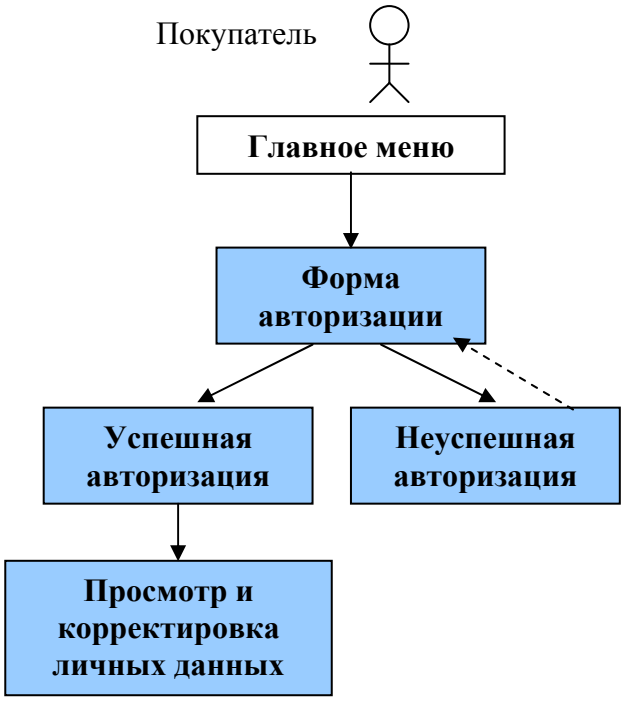
Мы будем использовать сессии для того, чтобы «опознавать» уже авторизованных покупателей.

Авторизация

Вспомним схему 6 со стр. 9 – авторизацию покупателя.

Для посетителей, зарегистрированных в нашем магазине, в дальнейшем не требуется заново вводить информацию о себе – достаточно просто набрать логин и пароль, т.е., авторизоваться. Для авторизации предназначена форма рядом с логотипом магазина. При нажатии на кнопку "ok" вызывается сценарий **auto.pthml**, в котором проверяется корректность введенных логина и пароля.

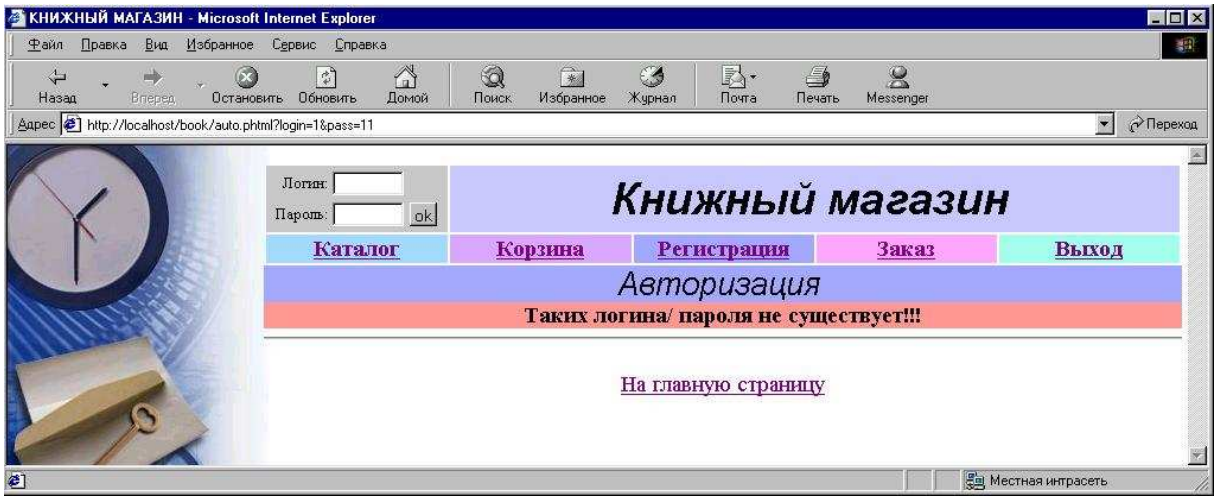
Схема 6. Авторизация




```
$pass=$HTTP_POST_VARS["pass"];
$login=$HTTP_POST_VARS["login"];

$strSQL1="SELECT * FROM customers WHERE login='".$login.
    "' AND pass='".$pass."'";
print $strsql;
$result1=mysql_query($strSQL1)
     or die("Не могу выполнить запрос!");

// пользователь с таким логином и паролем найден ?
if($row=mysql_fetch_array($result1))
{
   $start=session_start(); // начинаем сессию
   // создадим сеансовую переменную для ФИО покупателя
   session_register("log");
   $HTTP_SESSION_VARS["log"]=$row["fam"]." ".$row["im"];
   // создадим сеансовую переменную для ID покупателя
   session_register("id");
   $HTTP_SESSION_VARS["id"]=$row["id_cust"];

   $message="<tr><td bgcolor='#66cc66' align='center'>
      <b>Вы успешно авторизованы</b></td></tr>";
   $success=true;
}
else
{
   $message="<tr><td bgcolor='#ff9999' align='center'>
    <b>Таких логина/ пароля не существует!!!</b></td></tr>";
}
```

```
if($success)
{
   include ("cabinet.phtml");
}
else
{
   include("header.phtml");
   print $message;
   include("footer.phtml");
}
```

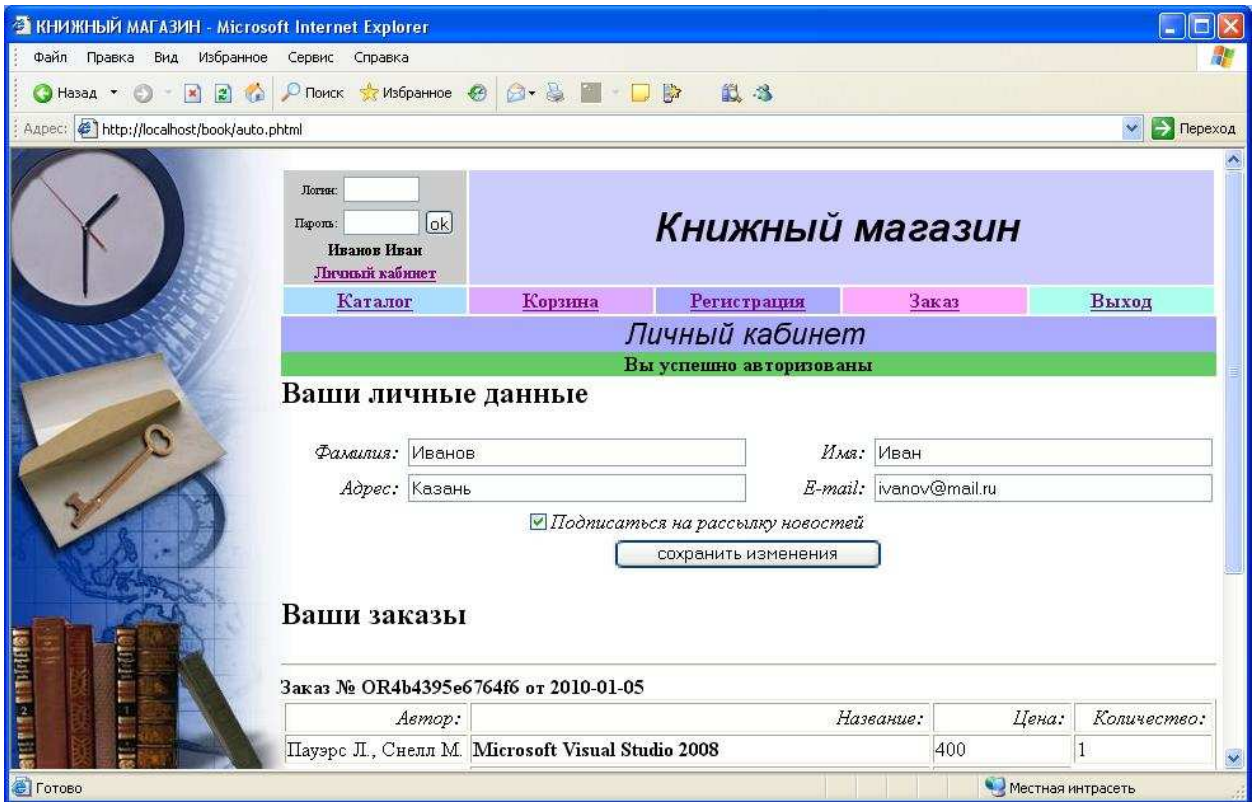
Если данные введены неверно, пользователь получит сообщение об ошибке:



Обратите внимание, что в случае успешной авторизации создаются сеансовые переменные с именами **id** (идентификатор) и **log** (фамилия, имя). Содержимое переменной **log** будет в дальнейшем выводиться под формой авторизации, а значение переменной **id** будет нужно для оформления заказа. В случае успешной авторизации перенаправляем пользователя в его «личный кабинет», где он получает возможность откорректировать информацию о себе и просмотреть свои заказы. Обратите внимание также, что в свой личный кабинет авторизованный пользователь сможет попасть в любой момент – для этого мы создадим специальную гиперссылку под формой авторизации (в файле **header.phtml**):

```
if(isset($HTTP_SESSION_VARS["log"]))
{
   print $HTTP_SESSION_VARS["log"];
```

```
   print "<br><a href='cabinet.phtml'>Личный кабинет</a>";
}
```



Сценарий личного кабинета пользователя, прежде всего, анализирует, существует ли сеансовая переменная **log** (т.е., авторизован ли пользователь)

```
$log=$HTTP_SESSION_VARS["log"];
$id=$HTTP_SESSION_VARS["id"];

if(!isset($log))
{
    $success=false;
    $message="<tr><td bgcolor='#ff9999' align='center'>
    <b>Вы не авторизованы!!!</b></td></tr>";
}
else
    $success=true;
```

Затем пользователь может, в частности, откорректировать свои данные

```
<form action=change.phtml method=post>
<tr><td>
<table border="0" width="100%" align="right" >
<tr><td align="right"><i>Фамилия: </i></td><td>
 <input type=text name=fam value="
   <?print $row["fam"]?>"></td></tr>
```

```
<tr><td align="right"><i>Имя: </i></td><td>
 <input type=text name=im value="
   <?print $row["im"]?>"></td></tr>
<tr><td align="right"><i>Адрес: </i></td><td>
 <input type=text name=addr value="
   <?print $row["addr"]?>"></td></tr>
<tr><td align="right"><i>E-mail: </i></td><td>
 <input type=text name=mail value="
   <?print $row["mail"]?>"></td></tr>
<tr><td align="right" colspan=3>
 <i><input type="checkbox" value="1" name="subscribe"
 <? if($row["subscribe"]==1) print "checked"; ?> >
 <i>Подписаться на рассылку новостей</i></i></td><td></td></tr>
<tr><td align="center" colspan="4">
  <input type="submit" value="сохранить изменения"></td></tr>
</table>
</form>
```

и просмотреть список своих старых заказов:

```
<tr><td>
<h2>Ваши заказы</h2>
<?
$strSQL1="SELECT id_order, date_ord FROM orders WHERE
id_cust='".$id."' ORDER BY date_ord DESC";
$result1=mysql_query($strSQL1) or die("Не могу выполнить
     запрос1!");
while($row1=mysql_fetch_array($result1))
{
  $order=$row1["id_order"];
  $strSQL2="SELECT author, name_book, pages, price, kolvo,
     id_order, books.id_book FROM books,   order_books
     WHERE books.id_book=order_books.id_book and
     id_order='".$order."'";
  $result2=mysql_query($strSQL2) or die("Не могу выполнить
     запрос2!");
  ?>
  <tr><td>
  <hr>
  <b>Заказ № <?=$order?> от <?=$row1["date_ord"]?><br></b>
  <table border="1" width="100%" align="right" >
  <tr><td align="right" width="20%"><i>Автор: </i></td>
     <td align="right" width="50%"><i>Название: </i></td>
     <td align="right" width="15%"><i>Цена: </i></td>
     <td align="right" width="15%"><i>Количество: </i></td>
```

```
  </tr>
  <?
  $sum=0;
  while($row2=mysql_fetch_array($result2))
  {
  ?>
    <tr>
     <td><?print $row2["author"];?></td>
     <td><b><?print $row2["name_book"];?></b></td>
     <td><?print $row2["price"];?></td>
     <td><?print $row2["kolvo"];?></td>
    </tr>
    <? $sum=$sum+$row2["price"]*$row2["kolvo"];
  }?>
  <tr><td></td><td align="right"><i>ИТОГО: </i></td>
  <td><? print $sum;?></td>
  <td></td></tr>
  </table>
  </td></tr>

  <?
}
?>
```

При нажатии на кнопку **"сохранить"** вызывается сценарий **change.phtml**, в котором производится изменение личных данных пользователя:

```
$fam=$HTTP_POST_VARS["fam"];
$im=$HTTP_POST_VARS["im"];
$addr=$HTTP_POST_VARS["addr"];
$mail=$HTTP_POST_VARS["mail"];
$id=$HTTP_SESSION_VARS["id"];
$subscribe=$HTTP_POST_VARS["subscribe"];

if($fam!="" && $im!="" && $addr!="" && $mail!="")
{
   $strSQL1="UPDATE customers SET fam='".$fam."',im='".$im.
   "',addr='".$addr."',mail='".$mail."',subscribe='".$subscribe.
   "' WHERE id_cust=".$id;
   $result1=mysql_query($strSQL1)
        or die("Не могу выполнить запрос!");
```

```
    $HTTP_SESSION_VARS["log"]=$fam." ".$im;
       // обновили значение сеансовой переменной
    $message="<tr><td bgcolor='#66cc66' align='center'>
       <b>Изменения данных выполнены</b></td></tr>";
}
else
$message="<tr><td bgcolor='#ff9999' align='center'>
    <b>Не все поля заполнены!!!</b></td></tr>";
```

Далее авторизованный пользователь сможет оформить заказ, а также продолжать работу с корзиной и просмотр каталога. Когда пользователь захочет выйти из магазина, ему следует щелкнуть по гиперссылке **"Выход"**. При этом будет вызван сценарий **exit.phtml**, основное действие которого заключается в завершении сессии пользователя:

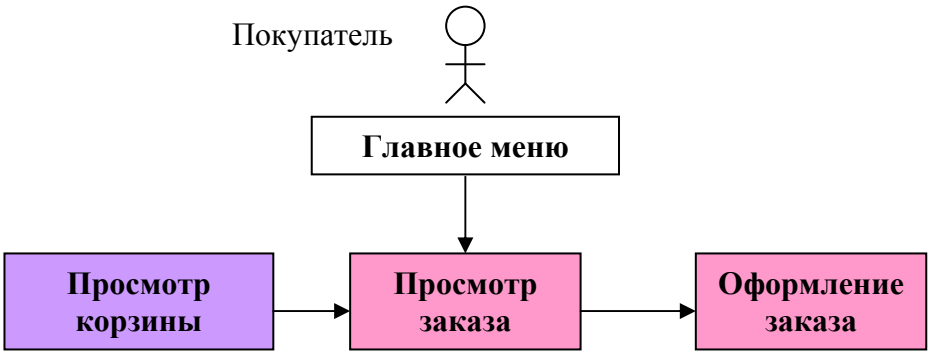
```
session_unregister("log");
session_unregister("id");
session_destroy();
```

Полностью коды сценариев приведены в Приложении 1.

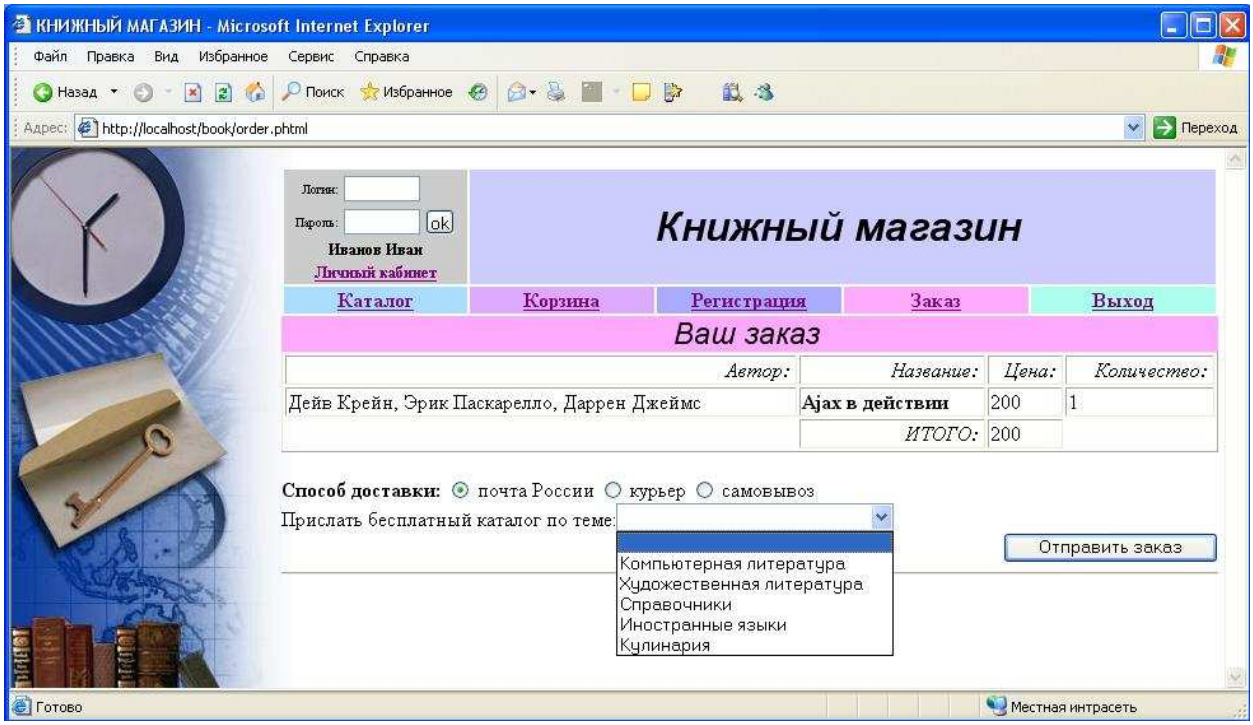
Этап 7. Разработка системы заказа

Вспомним схему 4 со стр. 8 – действия с заказом.

Схема 4. Работа с заказом



Когда пользователь решился совершить покупку, он должен выбрать пункт меню **"Заказ"** или щелкнуть по гиперссылке **"Оформить заказ"** в корзине. В обоих случаях будет вызван сценарий **order.phtml**, который позволит покупателю в последний раз проверить состав заказа (уже без возможности редактирования):



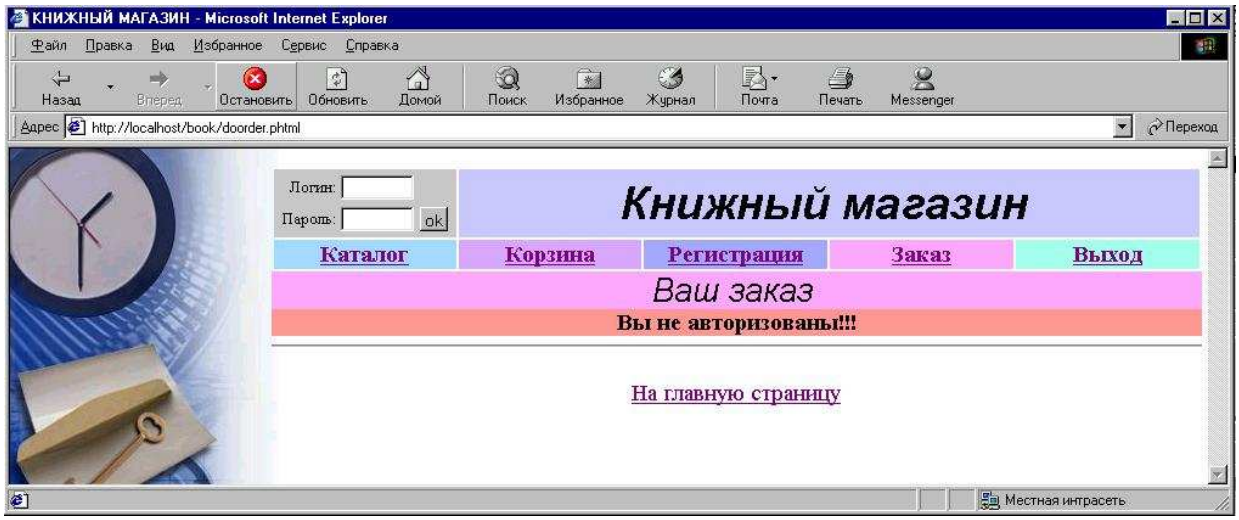
На этом этапе покупатель также должен выбрать тип доставки и, по желанию, бесплатный каталог по одной из категорий. Обратите внимание, как создаются элементы управления - радиокнопки и выпадающий список. Элементами списка являются названия категорий книг из таблицы **categories**.

```
<tr><td><br><b>Способ доставки:</b>
<input type="radio" value=1 name="dostavka" checked>
     почта России
<input type="radio" value=2 name="dostavka"> курьер
<input type="radio" value=3 name="dostavka"> самовывоз
</td><tr>
<tr><td>Прислать бесплатный каталог по теме:
     <select name="bonus">
     <option value="0">
     <? $strSQL1="SELECT * FROM categories";
        $result1=mysql_query($strSQL1)
            or die("Не могу выполнить запрос!");
        while($row=mysql_fetch_array($result1))
       {?>
        <option value="<? print $row["id_cat"]?>" >
           <? print $row["name_cat"]?>
       <?}?>
</td><tr>
```

Полностью код сценария приведен в Приложении 1.

При щелчке на гиперссылке **"Отправить заказ"** будет вызван сценарий **doorder.phtml**. Здесь, прежде всего, проверяется, авторизован ли покупатель:

```
$log=$HTTP_SESSION_VARS["log"];
$id=$HTTP_SESSION_VARS["id"];
$id_bask=$HTTP_COOKIE_VARS["id_bask"];

if(!isset($log))
      $message="<tr><td bgcolor='#ff9999' align='center'>
        <b>Вы не авторизованы!!!</b></td></tr>";
```



В том случае, когда покупатель авторизован, можно оформлять заказ. Оформление заказа с точки зрения данных заключается в следующих действиях.

Создается новый заказ для нашего покупателя в таблице Заказ:

```
$order=uniqid("OR");
$strSQL="INSERT INTO orders
 (id_order, date_ord, id_cust, dostavka, bonus) VALUES
 ('".$order."',CURDATE(),".$id.",'".$dostavka."','".$bonus."')";
mysql_query($strSQL)
    or die("Не могу выполнить запрос1!");
```

Выбираются строки из корзины покупателя и переписываются в таблицу СоставЗаказа:

```
$strSQL="SELECT * FROM basket_books
             WHERE id_bask='".$id_bask."'";
$result=mysql_query($strSQL)
               or die("Не могу выполнить запрос2!");
while ($row=mysql_fetch_array($result))
  {
   $strSQL="INSERT INTO order_books (id_order, id_book,
           kolvo) VALUES ('".$order."',".$row["id_book"].
           ",".$row["kolvo"].")";
   mysql_query($strSQL)
               or die("Не могу выполнить запрос3!");
  }
```

Удаляется корзина покупателя:

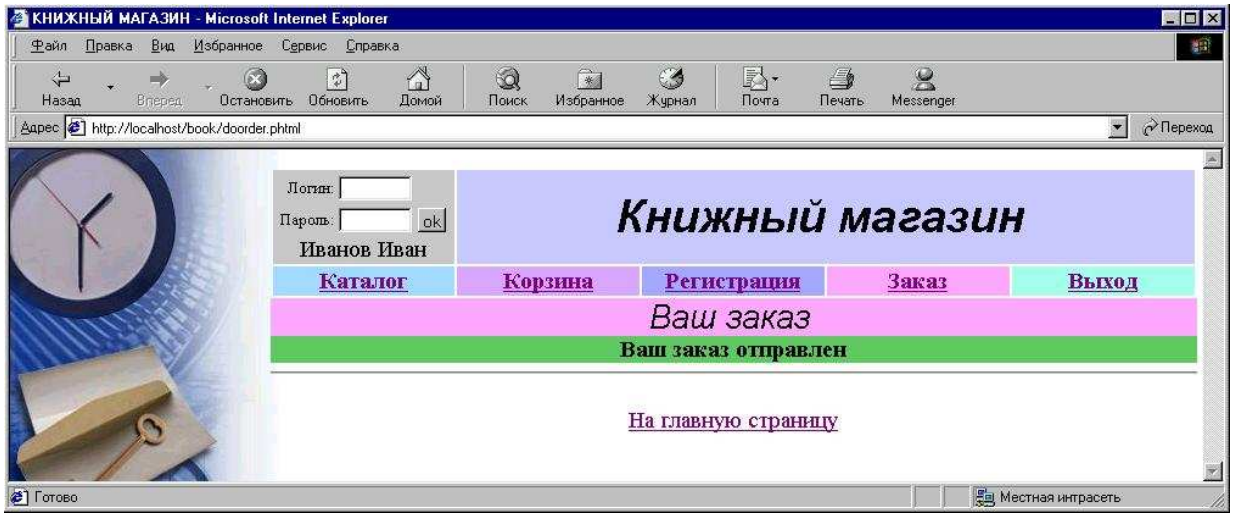
```
$strSQL="DELETE FROM basket_books
             WHERE id_bask='".$id_bask."'";
mysql_query($strSQL)
```

```
            or die("Не могу выполнить запрос4!");
$uniq_ID=uniqid("ID");
```

Создается новый идентификатор корзины:

```
setcookie("id_bask", $uniq_ID, time()+60*60*24*14);
```

Покупателю выдается сообщение об успешной отправке заказа:



Полностью коды сценариев приведены в Приложении 1.

Бонус: прайс-лист по издателям в формате XML

Каким образом можно создать следующий прайс-лист по всем книгам, сгруппированный по издателям, в формате **XML**? В примере приведен внешний вид **XML**-файла в браузере Internet Explorer.

```
<?xml version="1.0" encoding="windows-1251" ?>
- <прайс-лист>
- <издатель код="2">
    BHV
  - <книга Автор="Пауэрс Л., Снелл М." Название="Microsoft Visual Studio 2008">
      <страниц>1192</страниц>
      <цена>400</цена>
      <категория>Компьютерная литература</категория>
    </книга>
  - <книга Автор="Н. Культин" Название="Microsoft Visual C# в задачах и
    примерах">
      <страниц>314</страниц>
      <цена>140</цена>
      <категория>Компьютерная литература</категория>
    </книга>
  </издатель>
  <издатель код="5">Que</издатель>
- <издатель код="4">
    Диалектика
  - <книга Автор="Дейв Крейн, Эрик Паскарелло, Даррен Джеймс"
    Название="Ajax в действии">
      <страниц>640</страниц>
      <цена>200</цена>
      <категория>Компьютерная литература</категория>
    </книга>
  </издатель>
  <издатель код="3">ЛОРИ</издатель>
- <издатель код="1">
    Питер
  - <книга Автор="Бретт Маклафлин" Название="Изучаем Ajax">
      <страниц>425</страниц>
      <цена>300</цена>
      <категория>Компьютерная литература</категория>
    </книга>
  - <книга Автор="Д. Томас, Д. Х. Хэнссон" Название="Гибкая разработка веб-
    приложений в среде Rails">
      <страниц>720</страниц>
      <цена>400</цена>
      <категория>Компьютерная литература</категория>
    </книга>
</издатель>
</прайс-лист>
```

Прежде всего, следует сообщить браузеру, что дальнейший текст представляет собой **XML**-документ:

```
header ("Content-type: text/xml");
```

Первая строка любого **XML**-документа имеет стандартный вид и формируется следующим образом. Здесь мы явно указываем русскую кодировку Windows для того, чтобы корректно воспринимался русский текст.

```
print "<?xml version=\"1.0\" encoding=\"windows-1251\" ?>";
```

Далее выбираем из базы всех издателей и все книги по каждому издателю. К выбранным данным добавляем нужные тэги и выводим результат на печать:

```
$strSQL1="SELECT * FROM publishers ORDER BY name_publ";
$result1=mysql_query($strSQL1)
         or die("Не могу выполнить запрос1!");

print "<прайс-лист>";
while($row=mysql_fetch_array($result1))
{
  print "<издатель
код='".$row["id_publ"]."'>".$row["name_publ"];
  $strSQL2="SELECT id_book, author, name_book, pages, price,
            name_cat  FROM books, categories WHERE
                books.id_cat=categories.id_cat AND
                books.id_publ=".$row["id_publ"];
  $result2=mysql_query($strSQL2)
          or die("Не могу выполнить запрос2!");
  while($row2=mysql_fetch_array($result2))
  {
    print "<книга Автор='".$row2["author"].
                "' Название='".$row2["name_book"]. "'>";
    print "<страниц>".$row2["pages"]."</страниц>";
    print "<цена>".$row2["price"]."</цена>";
    print "<категория>".$row2["name_cat"]."</категория>";
    print "</книга>";
  }
  print "</издатель>";
}
print "</прайс-лист>";
```

Полностью код сценария приведен в Приложении 1.

Некоторые замечания

1. При загрузке некоторых страниц может случиться так, что браузер не захочет запрашивать страницу у сервера заново, а загрузит результат предыдущего выполнения того же сценария из собственного временного хранилища – из так называемого "кэша". Для того чтобы предотвратить такие ситуации, следует в серверном сценарии выполнить команду:

```
header("Cache-control: no-cache");
```

которая дает браузеру указание не сохранять результаты этой страницы в кэше. В нашем примере эта команда помещена в верхний шаблон.

2. Любой электронный магазин будет неполным без удобной системы поиска. Выполнив предыдущие семь заданий, вы уже обладаете всеми необходимыми знаниями для того, чтобы разработать эту систему самостоятельно.

3. Для дальнейшего совершенствования вашего магазина есть много направлений, которые выходят за рамки нашего краткого учебного пособия. Это и создание интерфейса для администратора, и подключение электронных платежных систем, и многое-многое другое.

Приложение 1. Коды сценариев

Шаблон для заголовка и меню (header.phtml)

```
<?
header("Cache-control: no-cache");

$id_bask=$HTTP_COOKIE_VARS["id_bask"];
if (! isset($id_bask))
{
     $uniq_ID=uniqid("ID");
     setcookie("id_bask", $uniq_ID, time()+60*60*24*14);
     // создадим ключик
}
else
     setcookie("id_bask", $id_bask, time()+60*60*24*14);
     // пересоздадим ключик с тем же значением

?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
     <title>КНИЖНЫЙ МАГАЗИН</title>
</head>

<body background="EULA.jpg" style="background-repeat:repeat-y"
leftmargin="130" rightmargin="5" bgProperties=fixed>
<table border="0" align="right" width="90%" cellpadding="0"
cellspacing="0">
<tr><td>

<table border="0" align="right" width="100%" >
<tr>
<td align="center" bgcolor="#cccccc">
<form action="auto.phtml" method="post">
<table>
<tr><td align="right"><font size=-2>Логин:</font></td>
<td align="left"><input type=text style="width:60; height:20;"
name=login></td></tr>
<tr><td align="right"><font size=-2>Пароль:</font></td>
<td align="left"><input type=password style="width:60;
height:20;" name=pass>
<input type=submit value=ok style="height:20;"></td></tr>
</table>
<b><small>
<?
if(isset($HTTP_SESSION_VARS["log"]))
{
   print $HTTP_SESSION_VARS["log"];
   print "<br><a href='cabinet.phtml'>Личный кабинет</a>";
}
```

```
?>
</small></b></td>
</form>
<td colspan="4" align="center" bgcolor="#ccccff">
<font face="Arial" size="+3"><i><b>Книжный
магазин</b></i></font></td></tr>
<tr><td align="center" bgcolor="#aaddff" width="20%">
<a href="catalog.phtml"><b>Каталог</b></a></td>
<td align="center" bgcolor="#ddaaff" width="20%">
<a href="basket.phtml"><b>Корзина</b></a></td>
<td align="center" bgcolor="#aaaaff" width="20%">
<a href="reg.phtml"><b>Регистрация</b></a></td>
<td align="center" bgcolor="#ffaaff" width="20%">
<a href="order.phtml"><b>Заказ</b></a></td>
<td align="center" bgcolor="#aaffee" width="20%">
<a href="exit.phtml"><b>Выход</b></a></td>
</tr>
</table>
</td></tr>
<tr><td align="center" bgcolor=<?print $color?>><font
face="Arial" size="+2">
<i><?print $title?></i></font><br>
</td></tr>
```

Шаблон для нижней части страницы (footer.phtml)

```
<tr><td><center><hr><br>
<a href="index.phtml">На главную страницу</a></center></td></tr>
</table>
</body>
</html>
```

Главная страница (index.phtml)

```
<?
$title="Welcome!";
$color="#ccccff";
include("header.phtml");
?>
<tr><td>
<center><h2><font color="#555599"><br>Добро пожаловать в наш
электронный книжный магазин! <br><br>Здесь представлена
литература самых известных российских издательств. Любой
читатель обязательно найдет себе книгу по вкусу! Мы предлагаем
компьютерную литературу, энциклопедии, справочники, самые
известные художественные произведения классиков и современников
и многое-многое другое!</font></h2></center>
</td></tr>
<? include("footer.phtml"); ?>
```

Шаблон для подключения к базе данных (connect.phtml)

```
<?
mysql_connect("localhost", "root", "") or
           die ("Не могу подключиться к  серверу!");
mysql_select_db("books") or
           die ("Не могу подключиться к базе данных!");
?>
```

Просмотр списка издателей и категорий (catalog.phtml)

```
<?
$title="Каталог";
$color="#aaddff";

include("header.phtml");
include("connect.phtml");

$strSQL1="SELECT * FROM publishers ORDER BY name_publ";
$result1=mysql_query($strSQL1)
     or die("Не могу выполнить запрос!");
$strSQL2="SELECT * FROM categories ORDER BY name_cat";
$result2=mysql_query($strSQL2)
     or die("Не могу выполнить запрос!");
?>
<tr><td>
<table border=0 width=100%>
<tr><td width="50%"><center><h3>Издатели</h3></center><ul>
<?
while($row=mysql_fetch_array($result1))
{?>
     <li><a href="show.phtml?type=1&id_publ=
         <?print $row["id_publ"];?>">
           <?print $row["name_publ"];?></a>
<?}?>
</ul></td>

<td width="50%"><center><h3>Категории</h3></center><ul>
<?
while($row=mysql_fetch_array($result2))
{?>
     <li><a href="show.phtml?type=2&id_cat=
         <?print $row["id_cat"];?>">
           <?print $row["name_cat"];?></a>
<?}?>
</ul></td>
</tr>
</table>
</td></tr>
<?
include("footer.phtml");
mysql_close();
?>
```

Просмотр списка книг по выбранному издателю или категории
(show.phtml)

```
<?
$id_publ=$HTTP_GET_VARS["id_publ"];
$id_cat=$HTTP_GET_VARS["id_cat"];
$type=$HTTP_GET_VARS["type"];

include("connect.phtml");

if ($type==1)
{
$strSQL1="SELECT name_publ FROM  publishers WHERE
id_publ=".$id_publ;
$result=mysql_query($strSQL1)
     or die("Не могу выполнить запрос1!");
if($row=mysql_fetch_array($result))
     $title=$row["name_publ"];
$strSQL1="SELECT id_book, image, author, name_book,
books.id_publ, name_publ, pages, price, books.id_cat, name_cat
FROM books, publishers, categories WHERE
books.id_cat=categories.id_cat AND
books.id_publ=publishers.id_publ AND books.id_publ=".$id_publ;
}
if ($type==2)
{
$strSQL1="SELECT name_cat FROM  categories WHERE
id_cat=".$id_cat;
$result=mysql_query($strSQL1)
     or die("Не могу выполнить запрос1!");
if($row=mysql_fetch_array($result))
     $title=$row["name_cat"];
$strSQL1="SELECT id_book, image, author, name_book,
books.id_publ, name_publ, pages, price, books.id_cat, name_cat
FROM books, publishers, categories WHERE
books.id_cat=categories.id_cat AND
books.id_publ=publishers.id_publ AND books.id_cat=".$id_cat;
}
$result1=mysql_query($strSQL1) or die("Не могу выполнить
запрос2!");

include("header.phtml");
?>
<tr><td>
<table border="1" width="100%" align="right" >
<?
while($row=mysql_fetch_array($result1))
{?>
  <tr>
  <td align="center"><img src="images/<?print $row["image"];?>"
      alt="<?print $row["name_book"];?>" border="0">
```

```
   <center><a href="dobasket.phtml?type=1&id_book=
       <?print $row["id_book"];?>">
       <font size=-1>положить в корзину</font></a></center></td>
  <td>
  <table>
  <tr><td align="right"><i>Автор: </i></td>
   <td><?print $row["author"];?></td></tr>
  <tr><td align="right"><i>Название: </i></td>
   <td><?print $row["name_book"];?></td></tr>
  <tr><td align="right"><i>Издательство: </i></td>
   <td><a href="show.phtml?type=1&id_publ=
    <?print $row["id_publ"];?>"><?print $row["name_publ"];?></a>
   </td></tr>
  <tr><td align="right"><i>Количество страниц: </i></td>
   <td><?print $row["pages"];?></td></tr>
  <tr><td align="right"><i>Цена: </i></td>
   <td><?print $row["price"];?></td></tr>
  <tr><td align="right"><i>Категория: </i></td>
   <td><a href="show.phtml?type=2&id_cat=
    <?print $row["id_cat"];?>"><?print $row["name_cat"];?></a>
   </td></tr>
  </table>
  </td>
  </tr>
<?}?>
</table>
</td></tr>
<?
include("footer.phtml");
?>
```

Просмотр корзины (basket.phtml)

```
<?
$id_bask=$HTTP_COOKIE_VARS["id_bask"];

$title="Ваша корзина";
$color="#ddaaff";
include("header.phtml");
include("connect.phtml");

$strSQL1="SELECT COUNT(*) as count FROM basket_books
     WHERE id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1)
     or die("Не могу выполнить запрос1!");
$row=mysql_fetch_array($result1);
if($row["count"]==0)
{
?>
     <tr><td bgcolor='#ff9999' align='center'>
        <b>Ваша корзина пуста!</b></td></tr>
<?
```

```
}
else
{
$strSQL1="SELECT image, author, name_book, pages, price, kolvo,
   id_bask, books.id_book FROM books, basket_books WHERE
   books.id_book=basket_books.id_book AND
id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1) or die("Не могу выполнить
запрос2!");
?>
<tr><td>
<table border="1" width="100%" align="right" >
<tr><td align="right"><i>Автор: </i></td>
<td align="right"><i>Название: </i></td>
<td align="right"><i>Цена: </i></td>
<td align="right"><i>Количество: </i></td>
<td></td></tr>
<?
$sum=0;
while($row=mysql_fetch_array($result1))
{
?>
  <tr>
  <td><?print $row["author"];?></td>
  <td><b><?print $row["name_book"];?></b></td>
  <td><?print $row["price"];?></td>
  <td><?print $row["kolvo"];?>
  <a href="dobasket.phtml?type=1&id_book=
     <?print $row["id_book"];?>" title="Увеличить">[ + ]</a>
  <a href="dobasket.phtml?type=2&id_book=
     <?print $row["id_book"];?>" title="Уменьшить">[ - ]</a>
  </td>
  <td> <a href="dobasket.phtml?type=3&id_book=
     <?print $row["id_book"];?>">Удалить</a></td>
</tr>
<?
  $sum=$sum+$row["price"]*$row["kolvo"];
}?>
<tr><td align="right"></td><td align="right"><i>ИТОГО:
</i></td><td align="right"><?print $sum;?></td><td
align="right"></td></tr>
</table>
<tr><td><center><a href=dobasket.phtml?type=4>
     <b>Очистить корзину</b></a></center></td></tr>
<tr><td><center><a href="order.phtml">
     <b>Оформить заказ</b></a></center></td></tr>
<?
}
include("footer.phtml");
?>
```

Действия с корзиной (dobasket.phtml)

```
<?
$type=$HTTP_GET_VARS["type"];
$id_book=$HTTP_GET_VARS["id_book"];
$id_bask=$HTTP_COOKIE_VARS["id_bask"];

include("connect.phtml");

if($type==1) // положить в корзину
{
     $strSQL="SELECT * FROM basket_books WHERE
           id_book=".$id_book." AND id_bask='".$id_bask."'";
     $result=mysql_query($strSQL)
           or die("Не могу выполнить запрос1!");
     if ($row=mysql_fetch_array($result))
     {
          $strSQL="UPDATE basket_books SET kolvo=kolvo+1 WHERE
              id_book=".$id_book." AND id_bask='".$id_bask."'";
     }
     else
     {
          $strSQL="INSERT INTO basket_books (id_bask, id_book,
                 kolvo) VALUES ('".$id_bask."',".$id_book.",1)";
     }
     mysql_query($strSQL);
}
else
if($type==2) // уменьшить количество
{
     $strSQL="SELECT * FROM basket_books WHERE
           id_book=".$id_book." AND id_bask='".$id_bask."'";
     $result=mysql_query($strSQL)
              or die("Не могу выполнить запрос1!");
     if ($row=mysql_fetch_array($result))
     {
          if ($row["kolvo"]>1)
          {
          $strSQL="UPDATE basket_books SET kolvo=kolvo-1 WHERE
             id_book=".$id_book." AND id_bask='".$id_bask."'";
          }
          else
          {
          $strSQL="DELETE FROM basket_books WHERE
             id_book=".$id_book." AND id_bask='".$id_bask."'";
          }
     }
     mysql_query($strSQL);

}
else
if($type==3) // удалить из корзины
{
```

```
     $strSQL="DELETE FROM basket_books WHERE
         id_book=".$id_book." AND id_bask='".$id_bask."'";
     mysql_query($strSQL);
}
else
if($type==4) // очистить корзину
{
     $strSQL="DELETE FROM basket_books WHERE
           id_bask='".$id_bask."'";
     mysql_query($strSQL);
}

include("basket.phtml");
?>
```

Регистрация (`reg.phtml`)

```
<?
$title="Регистрация";
$color="#aaaaff";

$fam=$HTTP_POST_VARS["fam"];
$im=$HTTP_POST_VARS["im"];
$addr=$HTTP_POST_VARS["addr"];
$mail=$HTTP_POST_VARS["mail"];
$pass=$HTTP_POST_VARS["pass"];
$pass2=$HTTP_POST_VARS["pass2"];
$login=$HTTP_POST_VARS["login"];
$type=$HTTP_POST_VARS["type"];
$subscribe=$HTTP_POST_VARS["subscribe"];

include("connect.phtml");
if($type==1)
{

if($fam!="" && $im!="" && $addr!="" && $mail!="" && $login!=""
            && $pass!="" && $pass2!="")
{
  if($pass!=$pass2)
  {
    $message="<tr><td bgcolor='#ff9999' align='center'><b>
    Поля пароля и повтора пароля не совпадают!!!</b></td></tr>";
  }
  else
  {
    $strSQL1="SELECT id_cust FROM customers
       WHERE login='".$login."'";
    $result1=mysql_query($strSQL1)
       or die("Не могу выполнить запрос!");
    if($row=mysql_fetch_array($result1))
     {
        $message="<tr><td bgcolor='#ff9999' align='center'>
```

```
        <b>Такой логин уже существует!!! Выберите другой
           логин</b></td></tr>";
      }
      else
      {
        $strSQL1="INSERT INTO customers
          (fam, im, addr, mail, login, pass, subscribe)
          VALUES('".$fam."','".$im."','".$addr."','".$mail.
           "','".$login."','".$pass."','".$subscribe."')";
        $result1=mysql_query($strSQL1)
           or die("Не могу выполнить запрос!");
           $message="<tr><td bgcolor='#66cc66' align='center'>
              <b>Вы успешно зарегистрированы</b></td></tr>";
        $success=true;
      }
    }
  }
else
  $message="<tr><td bgcolor='#ff9999' align='center'>
     <b>Не все поля заполнены!!!</b></td></tr>";
}

include("header.phtml");
print $message;

if(!$success)
{
?>

<form action=reg.phtml method=post>
<tr><td align="center">
<small>Звездочками отмечены обязательные поля</small>
<table border="0" width="100%" align="right" >
  <tr><td align="right" width="50%"><i>Фамилия: </i></td><td>
  <input type=text name=fam value="<?print $fam?>">*</td></tr>
  <tr><td align="right"><i>Имя: </i></td><td>
  <input type=text name=im value="<?print $im?>">*</td></tr>
  <tr><td align="right"><i>Адрес: </i></td><td>
  <input type=text name=addr value="<?print $addr?>">*</td></tr>
  <tr><td align="right"><i>E-mail: </i></td><td>
  <input type=text name=mail value="<?print $mail?>">*</td></tr>
  <tr><td align="right"><i>Логин: </i></td><td>
  <input type=text name=login value="<?print $login?>">*</td>
  </tr>
  <tr><td align="right"><i>Пароль: </i></td><td>
  <input type=password name=pass value="">*</td></tr>
  <tr><td align="right"><i>Повтор пароля: </i></td><td>
  <input type=password name=pass2 value="">*</td></tr>
  <tr><td></td><td>
  <input type="checkbox" value="1" name="subscribe">
```

```
   <i>Подписаться на рассылку новостей</i></td></tr>
   <input type=hidden value=1 name=type>
   <tr><td align="right"></td><td>
    <input type=submit value="отправить"></td></tr>
</table>
</form>
</td></tr>
<?
mysql_close();
}
include("footer.phtml");
?>
```

Авторизация (auto.phtml)

```
<?
$title="Авторизация";
$color="#aaaaff";

$pass=$HTTP_POST_VARS["pass"];
$login=$HTTP_POST_VARS["login"];

include("connect.phtml");

$strSQL1="SELECT * FROM customers WHERE login='".$login.
                "' AND pass='".$pass."'";
$result1=mysql_query($strSQL1)
          or die("Не могу выполнить запрос!");
if($row=mysql_fetch_array($result1))
{
  $start=session_start();
  session_register("log");
  $HTTP_SESSION_VARS["log"]=$row["fam"]." ".$row["im"];
  session_register("id");
  $HTTP_SESSION_VARS["id"]=$row["id_cust"];
  $message="<tr><td bgcolor='#66cc66' align='center'>
   <b>Вы успешно авторизованы</b></td></tr>";
  $success=true;
}
else
{
  $message="<tr><td bgcolor='#ff9999' align='center'>
  <b>Таких логина/ пароля не существует!!!</b></td></tr>";
}
mysql_close();

if($success)
{
   include ("cabinet.phtml");
}
else
{
```

```
   include("header.phtml");
   print $message;
   include("footer.phtml");
}
?>
```

Личный кабинет (cabinet.phtml)

```
<?
$title="Личный кабинет";
$color="#aaaaff";

$log=$HTTP_SESSION_VARS["log"];
$id=$HTTP_SESSION_VARS["id"];

if(!isset($log))
{
  $success=false;
  $message="<tr><td bgcolor='#ff9999' align='center'>
     <b>Вы не авторизованы!!!</b></td></tr>";
}
else
  $success=true;

include("header.phtml");
print $message;

if($success)
{
include("connect.phtml");
$strSQL="SELECT * FROM customers WHERE id_cust='".$id."'";
$result=mysql_query($strSQL)
           or die("Не могу выполнить запрос!");
if($row=mysql_fetch_array($result))
{
?>
<form action=change.phtml method=post>
<tr><td>
<h2>Ваши личные данные</h2>
<table border="0" width="100%" align="right" >
 <tr><td align="right"><i>Фамилия: </i></td><td>
 <input type=text name=fam value="<?print $row["fam"]?>"></td>
 <td align="right"><i>Имя: </i></td><td>
 <input type=text name=im value="<?print $row["im"]?>"></td>
 </tr>
 <tr><td align="right"><i>Адрес: </i></td><td>
 <input type=text name=addr value="<?print $row["addr"]?>"></td>
 <td align="right"><i>E-mail: </i></td><td>
 <input type=text name=mail value="<?print $row["mail"]?>"></td>
 </tr>
 <tr><td align="right" colspan=3><i>
 <input type="checkbox" value="1" name="subscribe"
 <? if($row["subscribe"]==1) print "checked"; ?> >
```

```
 <tr><td align="center" colspan="4">
 <input type="submit" value="сохранить изменения"></td></tr>
</table>
</form>
</td></tr>
<tr><td>
<h2>Ваши заказы</h2>
<?
$strSQL1="SELECT id_order, date_ord FROM orders
     WHERE id_cust='".$id."' ORDER BY date_ord DESC";
$result1=mysql_query($strSQL1)
          or die("Не могу выполнить запрос1!");
while($row1=mysql_fetch_array($result1))
{
  $order=$row1["id_order"];
  $strSQL2="SELECT author, name_book, pages,
     price, kolvo, id_order, books.id_book
     FROM books,   order_books WHERE
     books.id_book=order_books.id_book
     and id_order='".$order."'";
  $result2=mysql_query($strSQL2)
          or die("Не могу выполнить запрос2!");
  ?>
  <tr><td>
  <hr>
  <b>Заказ № <?=$order?> от <?=$row1["date_ord"]?><br></b>
  <table border="1" width="100%" align="right" >
  <tr><td align="right" width="20%"><i>Автор: </i></td>
  <td align="right" width="50%"><i>Название: </i></td>
  <td align="right" width="15%"><i>Цена: </i></td>
  <td align="right" width="15%"><i>Количество: </i></td></tr>
  <?
  $sum=0;
  while($row2=mysql_fetch_array($result2))
  {
  ?>
    <tr>
     <td><?print $row2["author"];?></td>
     <td><b><?print $row2["name_book"];?></b></td>
     <td><?print $row2["price"];?></td>
     <td><?print $row2["kolvo"];?></td>
    </tr>
  <?$sum=$sum+$row2["price"]*$row2["kolvo"];
  }
  $strSQL3="SELECT name_cat FROM categories, orders
    WHERE categories.id_cat=orders.bonus AND
          id_order='".$order."'";
  $result3=mysql_query($strSQL3)
        or die("Не могу выполнить запрос3!");
  if($row3=mysql_fetch_array($result3))
  {?>
```

```
  <tr>
    <td colspan=2>Бесплатный каталог по теме <b>
    <?print $row3["name_cat"];?></b></td>
    <td>0</td>
    <td>1</td>
   </tr>
  <?}
  ?>
  <tr><td></td><td align="right"><i>ИТОГО: </i></td>
   <td><?print $sum;?></td><td></td></tr>
  </table>
  </td></tr>

  <?
  }
}
mysql_close();
}

include("footer.phtml");
?>
```

Изменение личных данных (change.phtml)

```
<?
$fam=$HTTP_POST_VARS["fam"];
$im=$HTTP_POST_VARS["im"];
$addr=$HTTP_POST_VARS["addr"];
$mail=$HTTP_POST_VARS["mail"];
$id=$HTTP_SESSION_VARS["id"];
$subscribe=$HTTP_POST_VARS["subscribe"];

$title="Регистрация";
$color="#aaaaff";

include("connect.phtml");

if($fam!="" && $im!="" && $addr!="" && $mail!="")
{
    $strSQL1="UPDATE customers SET fam='".$fam."',
        im='".$im."',addr='".$addr."', mail='".$mail."',
        subscribe='".$subscribe."' WHERE id_cust=".$id;
    $result1=mysql_query($strSQL1)
        or die("Не могу выполнить запрос!");
    $HTTP_SESSION_VARS["log"]=$fam." ".$im;
        // обновили значение сеансовой переменной
    $message="<tr><td bgcolor='#66cc66' align='center'>
       <b>Изменения данных выполнены</b></td></tr>";
}
else
$message="<tr><td bgcolor='#ff9999' align='center'>
    <b>Не все поля заполнены!!!</b></td></tr>";
```

```
include("header.phtml");
print $message;

include("footer.phtml");
?>
```

Выход – отмена авторизации (exit.phtml)

```
<?
session_unregister("log");
session_unregister("id");
session_destroy();

include("index.phtml");
?>
```

Просмотр заказа (order.phtml)

```
<?
$id_bask=$HTTP_COOKIE_VARS["id_bask"];

$title="Ваш заказ";
$color="#ffaaff";
include("header.phtml");
include("connect.phtml");

$strSQL1="SELECT COUNT(*) as count FROM basket_books
     WHERE id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1)
     or die("Не могу выполнить запрос2!");
$row=mysql_fetch_array($result1);
if($row["count"]==0)

{
?>
<tr><td bgcolor='#ff9999' align='center'>
  <b>Ваша корзина пуста!</b></td></tr>
<?
}
else
{
$strSQL1="SELECT image, author, name_book, pages, price, kolvo,
   id_bask, books.id_book FROM books, basket_books
  WHERE books.id_book=basket_books.id_book AND
   id_bask='".$id_bask."'";
$result1=mysql_query($strSQL1)
   or die("Не могу выполнить запрос2!");

?>
<tr><td>
<table border="1" width="100%" align="right" >
```

```
<tr><td align="right"><i>Автор: </i></td>
<td align="right"><i>Название: </i></td>
<td align="right"><i>Цена: </i></td>
<td align="right"><i>Количество: </i></td></tr>
<?
$sum=0;
while($row=mysql_fetch_array($result1))
{
?>
  <tr>
   <td><?print $row["author"];?></td>
   <td><b><?print $row["name_book"];?></b></td>
   <td><?print $row["price"];?></td>
   <td><?print $row["kolvo"];?></td>
  </tr>
<?
  $sum=$sum+$row["price"]*$row["kolvo"];
}
?>
<tr><td></td><td align="right"><i>ИТОГО: </i></td>
<td><?print $sum;?></td><td></td></tr>
</table>
<tr><td><br><b>Способ доставки:</b>
  <input type="radio" value=1 name="dostavka" checked>
   почта России
  <input type="radio" value=2 name="dostavka"> курьер
  <input type="radio" value=3 name="dostavka"> самовывоз
</td><tr>
<tr><td>Прислать бесплатный каталог по теме:
  <select name="bonus">
  <option value="0">
  <? $strSQL1="SELECT * FROM categories";
     $result1=mysql_query($strSQL1)
        or die("Не могу выполнить запрос!");
     while($row=mysql_fetch_array($result1))
     {?>
      <option value="<? print $row["id_cat"]?>">
      <? print $row["name_cat"];
     }
  ?>
</td><tr>
<tr><td><center><a href=doorder.phtml>
   <b>Отправить заказ</b></a></center></td></tr>
<?
}
include("footer.phtml");
?>
```

Отправка заказа (doorder.phtml)

```
<?
```

```
$log=$HTTP_SESSION_VARS["log"];
$id=$HTTP_SESSION_VARS["id"];
$id_bask=$HTTP_COOKIE_VARS["id_bask"];
$dostavka=$HTTP_POST_VARS["dostavka"];
$bonus=$HTTP_POST_VARS["bonus"];

$title="Ваш заказ";
$color="#ffaaff";

include("connect.phtml");

if(!isset($log))
       $message="<tr><td bgcolor='#ff9999' align='center'>
         <b>Вы не авторизованы!!!</b></td></tr>";
else
{
     $strSQL1="SELECT COUNT(*) as count FROM basket_books
        WHERE id_bask='".$id_bask."'";
     $result1=mysql_query($strSQL1)
        or die("Не могу выполнить запрос2!");
     $row=mysql_fetch_array($result1);
     if($row["count"]==0)
        $message="<tr><td bgcolor='#ff9999' align='center'>
           <b>Ваша корзина пуста!</b></td></tr>";
     else
     {
        // создаем новый заказ
        $order=uniqid("OR");
        $strSQL="INSERT INTO orders
          (id_order, date_ord, id_cust, dostavka, bonus)
          VALUES ('".$order."',CURDATE(),".$id.",
          '".$dostavka."','".$bonus."')";
        mysql_query($strSQL)
              or die("Не могу выполнить запрос1!");
        // читаем все из корзины покупателя
        $strSQL="SELECT * FROM basket_books
             WHERE id_bask='".$id_bask."'";
        $result=mysql_query($strSQL)
              or die("Не могу выполнить запрос2!");
        while ($row=mysql_fetch_array($result))
        {
          // и переписываем в состав заказа
          $strSQL="INSERT INTO order_books (id_order, id_book,
              kolvo) VALUES ('".$order."',".$row["id_book"].
              ",".$row["kolvo"].")";
          mysql_query($strSQL)
               or die("Не могу выполнить запрос3!");
        }
        // очищаем корзину покупателя
        $strSQL="DELETE FROM basket_books
             WHERE id_bask='".$id_bask."'";
        mysql_query($strSQL)
```

```
            or die("Не могу выполнить запрос4!");
        $uniq_ID=uniqid("ID");
        setcookie("id_bask", $uniq_ID, time()+60*60*24*14);
        $message="<tr><td bgcolor='#66cc66' align='center'>
           <b>Ваш заказ отправлен</b></td></tr>";
  }
}
include("header.phtml");
print $message;
include("footer.phtml"); ?>
```

Прайс-лист в формате XML (`price.phtml`)

```
<?
header ("Content-type: text/xml");
print "<?xml version=\"1.0\" encoding=\"windows-1251\" ?>";

include("connect.phtml");
$strSQL1="SELECT * FROM publishers ORDER BY name_publ";
$result1=mysql_query($strSQL1) or die("Не могу выполнить
запрос1!");

print "<прайс-лист>";
while($row=mysql_fetch_array($result1))
{
  print "<издатель
код='".$row["id_publ"]."'>".$row["name_publ"];
  $strSQL2="SELECT id_book, author, name_book, pages, price,
name_cat  FROM books, categories WHERE
books.id_cat=categories.id_cat AND
books.id_publ=".$row["id_publ"];
  $result2=mysql_query($strSQL2) or die("Не могу выполнить
запрос2!");
  while($row2=mysql_fetch_array($result2))
  {
    print "<книга Автор='".$row2["author"]."'
Название='".$row2["name_book"]. "'>";
    print "<страниц>".$row2["pages"]."</страниц>";
    print "<цена>".$row2["price"]."</цена>";
    print "<категория>".$row2["name_cat"]."</категория>";
    print "</книга>";
  }
  print "</издатель>";
}
print "</прайс-лист>";
mysql_close();
?>
```

Приложение 2. Некоторые аспекты технологии PHP

Регулярные выражения

Регулярные выражения, или, что то же самое – строковые шаблоны – пришли в **PHP** из **UNIX** и **Perl**. Эта тема настолько обширна, что по ней написаны целые книги, например, ***Фридл, Регулярные выражения. Библиотека программиста - Питер, 2001.*** Поэтому здесь мы рассмотрим только самые основы философии регулярных выражений. В **PHP** используется два формата регулярных выражений - в стиле **POSIX** и в стиле **Perl**.

Стиль POSIX

- | - вертикальная черта - означает выбор одной из альтернатив, например, **"cat|dog"**.
- **[]** - означают любой символ из перечисленных в скобках, например **"[abcd]"**. Для краткости можно задавать интервал в форме **[0-9]** или **[a-z]**. Если в скобках встречается символ ^ - это означает отрицание, например, **[^0-9]** означает 'любой символ, кроме цифр'. Для некоторых интервалов заданы псевдонимы:
  - **[[:alpha:]]** эквивалентно **[a-zA-Z]**,
  - **[[:alnum:]]** эквивалентно **[a-zA-Z0-9]**,
  - **[[:digit:]]** эквивалентно **[0-9]**,
  - **[[:space]]** - пробел, \n, \t и т.п.
- +, *, **?** и **{...}** - квантификаторы, означающие определенное количество повторов подстроки. Например,
  - "**a+**" - один или несколько подряд идущих символов 'a',
  - "**a***" - ноль или более подряд идущих символов 'a',
  - "**a?**" - ноль или один символ 'a',
  - "**a{3}**" - три подряд идущих символов 'a'
  - "**a{3,}**" - три или более подряд идущих символов 'a'.
- **$** - означает конец строки, например, **"ский$"** - строка, заканчивающаяся на 'ский'.
- ^ - означает начало строки, например, **"^Нью"** - строка, начинающаяся на 'Нью'.

- **.** - означает любой символ.
- **()** - круглые скобки - отделяют одно подвыражение в шаблоне от другого.

Если нужно искать в строке символ, совпадающий со служебным, то его экранируют с помощью обратной косой черты '\'.

С помощью отдельных служебных символов можно строить выражения любой сложности. Например,

- **"<h1>(.*)</h1>"** - строка, отформатированная как HTML-заголовок первого уровня.
- **"(\.)(ru$)"** - строка, оканчивающаяся на **.ru**
- **"(^http://)(.*)(w{3})"** - строка, начинающаяся на 'http://' и содержащая 'www'.
- **"^.{10}$"** - строка длиной 10.

Для работы с **POSIX**- шаблонами служат следующие функции:

int ereg (string шаблон, string строка [, array совпадения])	возвращает истину, если строка соответствует шаблону. Если задан параметр 'совпадения', то нулевой элемент этого массива будет содержать всю строку, а остальные элементы - совпадения для подвыражений в круглых скобках
string ereg_replace (string шаблон, string замена, string строка)	ищет в строке совпадения по шаблону и, если найдет, заменяет новой подстрокой
int eregi (string шаблон, string строка [, array совпадения])	эквивалентно **ereg**, но без учета регистра символов
string eregi_replace (string шаблон, string замена, string строка)	эквивалентно **ereg_replace**, но без учета регистра символов
array split (string шаблон, string строка [, int порог])	расщепляет строку на подстроки, границы которых определяются по заданному шаблону (похоже на **explode**)
array spliti (string шаблон, string строка [, int порог])	эквивалентно **split**, но без учета регистра символов

Объекты

Работа с объектами в **PHP** реализована на очень примитивном уровне. Все-таки язык **PHP** - процедурный, и его объектно-ориентированные возможности представляют собой не более чем небольшое архитектурное украшение. Впрочем, для простых задач этого вполне достаточно.

Реально в **PHP** реализованы только принципы **инкапсуляции** и простого **наследования**. Отсутствуют:

- перегрузка методов,
- управление доступом,
- виртуальные функции и полиморфизм,
- абстрактные классы,
- множественное наследование,
- деструкторы,
- неявный вызов конструкторов.

Краткое описание объектов см. в главе "Типы данных". Определение производного класса напоминает синтаксис Java:

```
class Animal
{ var $color, $weight, $height;
  function Animal($c, $w, $h)
  {
     $this->color=$c;
     $this->weight=$w;
     $this->height=$h;
  }
}

class Bird extends Animal
{ var $isSongBird,   // певчая птица
  $isMigrantBird;    // перелетная птица
  function Bird($c, $w, $h, $isS, $isM)
  {
     $this->isSongBird=$isS;
     $this->isMigrantBird=$isM;
     $this->Animal($c, $w, $h);
  }
}
```

Есть несколько полезных функций для работы с классами.

array get_class_methods (string имя_класса)	возвращает массив имен методов класса

array get_class_vars (string имя_класса)	возвращает массив имен атрибутов класса
array get_object_vars (object имя_объекта)	возвращает массив имен атрибутов объекта
bool method_exists (object имя_объекта, string имя_метода)	проверяет, есть ли у объекта метод с таким именем
string get_class (object имя_объекта)	возвращает имя класса
string get_parent_class (object имя_объекта)	возвращает имя родительского класса
bool is_subclass_of (object имя_объекта, string имя_класса)	проверяет, является ли объект производным от данного родительского класса

Файлы

Очень легко в **PHP** работать с файлами – почти так же, как в **C**. Здесь только нет различия между текстовыми и двоичными файлами (практически всегда используется именно текстовая информация). Действия с файлами выполняются, разумеется, с помощью функций.

Проверка существования и размера файла	
bool file_exists (string файл)	проверяет, существует ли файл
bool is_file (string файл)	проверяет, существует ли файл и можно ли его читать/писать
bool is_readable (string файл)	проверяет, существует ли файл и можно ли его читать
bool is_writable (string файл)	проверяет, существует ли файл и можно ли его писать
int filesize (string файл)	возвращает размер файла
Открытие/закрытие файла	

int fopen (string файл, string режим [, int включение_пути])	открывает файл - он может быть локальным, может представлять собой стандартный поток (**php://stdin, php://stdout, php://stderr**) или удаленный файл (с префиксами **http://** или **ftp://**). Режимы открытия такие же, как в **C** (**r, r+, w, w+, a, a+**). Возвращает файловый манипулятор, который используется в дальнейшем для чтения/записи.
int fclose (int манипулятор)	закрывает файл
Чтение файла	
bool feof (int манипулятор)	проверяет, не достигнут ли конец файла
string fread (int манипулятор, int длина)	читает из файла и возвращает заданное количество байт
string fgetc (int манипулятор)	читает из файла и возвращает один байт
string fgets (int манипулятор, int длина)	читает из файла и возвращает строку; чтение прекращается, если достигнут конец файла или конец строки, или если прочитано заданное количество байт
string fgetss (int манипулятор, int длина [, string разрешенные_теги])	также читает строку, а кроме того, удаляет из нее все **HTML** и **PHP**-теги, кроме разрешенных
array file (string файл)	читает из файла все строки и помещает их в массив (заметьте, что файл не нужно открывать!)
int readfile (string файл [, int включение_пути])	читает из файла всю информацию и перенаправляет ее в стандартный поток вывода (как правило, в браузер)
Запись в файл	
int fwrite (int манипулятор, string переменная [,int длина])	записывает строку в файл, запись прекращается, если достигнут конец строки или если записано заданное количество байт
int fputs (int манипулятор, string переменная [,int длина])	записывает строку в файл, запись прекращается, если достигнут конец строки или если записано заданное количество байт

Кроме того, есть функции для работы с файловой системой - для копирования, переименования, удаления, изменения прав доступа файлов, для манипулирования каталогами и пр. Для примера приведем функцию запуска внешних программ.

string exec (string команда [, array массив [,int возврат]])	выполняет команду операционной системы и возвращает последнюю строку ее выходных данных. Также можно (но необязательно) получить в виде массива **все** выходные данные и код возврата целого типа.

Конфигурация PHP

Параметры конфигурации **PHP** задаются в файле инициализации **php.ini**, который, как вы помните, мы поместили в каталог **WINDOWS** при установке **PHP**.

Этот файл разбит на отдельные секции, в которых сгруппированы параметры. Точка с запятой означает однострочный комментарий. Большинство параметров можно принять по умолчанию, но, разумеется, при создании более-менее сложных сценариев иногда приходится настраивать PHP под свои задачи.

Вспомним, что мы уже производили некоторые настройки в этом файле. Режим регистрации глобальных переменных можно включать или отключать командой

```
register_globals = on | off
```

Для автостарта сессий следует установить параметр:

```
session.auto_start = 1
```

Рассмотрим прочие наиболее часто используемые параметры конфигурации.

```
short_open_tags = on | off
```

разрешает/запрещает использование коротких тегов **<? ... ?>** для сценариев. Если он отключен, то можно использовать только теги **<?php ... ?>** и **<script> ... </script>** .

asp_open_tags = on | off

разрешает или запрещает использование тегов **<% ... %>** для сценариев.

precision = целое_число

определяет количество значащих цифр в вещественных числах.

max_execution_time = целое_число

задает тайм-аут – максимальный интервал времени (в секундах) для выполнения сценария. По истечении этого интервала выполнение прерывается, что позволяет останавливать зависшие сценарии.

memory_limit = целое_число

определяет максимальный объем памяти, отводимый для сценария (по умолчанию - 8M).

error_reporting = 1 | 2 | 4 | 8

задает уровень чувствительности **PHP** к ошибкам - от обычных ошибок (1) до предупреждений (8).

display_errors = on | off

управляет выводом информации об ошибках.

error_log = имя_файла

задает имя файла для сбора информации об ошибках при включенном **display_errors**.

magic_quotes_gpc = on | off

при включенном параметре **все** управляющие символы внутри строк автоматически экранируются обратной косой чертой (например, символы кавычек).

safe_mode = on | off

позволяет включить "безопасный" режим работы, который сильно ограничивает возможности сценариев.

safe_mode_exec_dir = имя_каталога

задает при включенном **safe_mode** каталог для размещения системных программ, запускаемых, например, функцией **system()**.

disable_functions = список_функций

задает список функций, запрещенных для выполнения.

Специфическая настройка конфигурации требуется, в частности, при создании изображений средствами PHP. Для того чтобы работать с графикой, необходимо сначала подключить соответствующую библиотеку функций. Дело в том, что графические функции не подключаются автоматически при использовании PHP, в отличии, например, от функций для работы с файлами и строками. Для подключения библиотеки следует открыть файл **php.ini**, найти секцию **Paths and Directories** и установить в качестве значения параметра **extension_dir** имя каталога с расширениями PHP, например,

extension_dir = C:/PHP/Extensions

Кроме того, нужно раскомментировать строку

;extension=php_gd.dll

т.е., убрать точку с запятой в начале строки. Все готово для работы с графикой.

Приложение 3. Настройка PHP под MS IIS

Технология **PHP** может работать не только в рамках сервера **Apache**. Достаточно легко настроить **PHP** и под сервер **Microsoft Internet Information Services** (операционные системы **Windows NT**, **2000** или **XP**).

Во-первых, запустите **IIS** и попробуйте загрузить в браузер какой-либо **PHTML**-файл. Скорее всего, вы просто увидите текст сценария. Это означает, что **PHP** пока не настроен.

Рассмотрим, как выглядит процесс настройки для **Windows XP**. Для этого выберите **«Пуск» - «Панель управления» - «Администрирование» - «Internet Information Services»**, щелкните правой кнопкой по строке **«Web-узел по умолчанию»** и выберите пункт **«Свойства»**. Вы увидите окно настройки **IIS**. Перейдите на закладку **«Домашний каталог»** и нажмите кнопку **«Настройка»**. Вы увидите список типов файлов, которые понимает **IIS**. Нажмите кнопку **«Добавить»**. В качестве исполняемого файла выберите полный путь к файлу **php.exe** – он может выглядеть, например, как **C:\php\php.exe**. В поле **«Расширение»** наберите **.phtml**. Закройте окна настройки.

Так же, как и для настройки под Apache, требуется поместить файл конфигурации **php.ini** в каталог **C:\WINDOWS,** а файлы **php4ts.dll** и **Msvcrt.dll –** в каталог **C:\WINDOWS\SYSTEM32**.

Теперь снова попробуйте загрузить в браузер какой-либо **PHTML**-файл. Вы увидите результаты его работы.

Разумеется, эта настройка производится один раз.

Примечание. При работе с IIS могут возникнуть проблемы с сеансовыми переменными. В файле php.ini следует указать имя каталога для хранения сессий, например:

```
session.save_path = C:\tmp
```

Примечание. Для некоторых версий PHP может потребоваться явная установка параметра:

```
cgi.force_redirect = 0
```

Литература

1. Водолазкий В. - **Эффективная работа: PHP 4.** Изд-во Питер, 2002.
2. Гилмор В. - **PHP4.Учебный курс.** Изд-во Питер, 2001.
3. Косентино К. - **PHP для Web-профессионалов.** БХВ-Киев, 2001.
4. Костарев А. - **PHP в Web-дизайне** БХВ-Петербург, 2002.
5. Ратшиллер Т., Геркен Т. - **PHP4. Разработка Web-приложений.** Изд-во Питер, 2001.
6. Томсон Л. - **Разработка Web-приложений на PHP и MySQL.** ДиаСофт, 2001.
7. Фролов А. - **Практика применения PERL, PHP, APACHE и MySQL для активных Web-сайтов.** Изд-во Русская редакция, 2002.

Для заметок