Интернет-магазин средствами MySQL & PHP

# Магазин на экране...

## (часть 2)

В прошлом номере мы рассказывали о том, как создать простой интернет-магазин, взяв за основу объектно-ориентированный подход. Это, с одной стороны, позволило строить приложение из отдельных блоков, как из кубиков ЛЕГО, с другой стороны — отделить работу программиста-разработчика функциональной части от работы web-дизайнера. Сегодня мы продолжим рассказ о построении онлайнового магазина и рассмотрим, как сформировать корзину покупок и отправить заказ продавцу по электронной почте. Все листинги и первую часть статьи в формате PDF, вы найдете на Chip CD.

Итак, нами построен иерархический каталог, по которому покупатель выбирает товары интересующей его рубрики. Понятие «иерархический» означает, что каталог организован по принципу дерева так же, как организованы файловые системы в Windows и Unix. Для того чтобы не перегружать код примера сложными процедурами на JavaScript, отображающими дерево, которые легко найти в Интернете, мы будем показывать только текущий уровень каталога со ссылкой на уровень более высокий (рис. 1). В правом фрейме выводится список товаров, принадлежащих этой рубрике. Скрипт, который реализует каталог (listcatalog.php), передает код выбранной рубрики в правый фрейм через переменную $idtype.

## Выбор товаров

Выбор товаров в корзину организован в виде формы, которая содержит поля ввода желаемого количества. Здесь нет ничего сложного, лишь выводится список записей, которые в поле idGoods содержат код типа товара, переданный в качестве параметра $idtype.

Класс Goods определен в том же файле, что и каталог товаров, — catalog.php. В нашей упрощенной версии магазина он довольно прост и содержит только методы, которые выводят список товаров. Большего от него в нашем примере не требуется.

Как уже говорилось в первой части, желательно отделить функции программиста от функций дизайнера. Программист обеспечивает работу магазина, а дизайнер — его внешний вид. Чтобы их работа зависела друг от друга минимально, мы воспользовались механизмом виртуальных методов в языке PHP. Класс Goods, как и остальные классы — Catalog и Orders, разрабатывается программистом и не содержит тэгов HTML. Список товаров выводится методом list_goods. Он выполняет запрос к базе данных и вызывает абстрактные методы print_form_header(), print_item() и print_footer(). Сами эти методы не содержат вывода HTML-тэгов, поэтому и называются абстрактными. Они лишь определяют, как и с какими параметрами их надо вызывать. В файле listgoods.php создается класс-потомок от Goods с именем Order. Перечисленные в нем методы print_* дизайнер заполняет конкретными тэгами, где нужно подставляя значения параметров.

Механизм заказа здесь прост: товары по данной рубрике выводятся в форму и покупатель вводит требуемое количество. Рассмотрим построение самой формы в классе-потомке, который определяется в файле listgoods.php:

```
<form method='post' action='listorder.php?u-
ser_session=$user_session&idtype=$idtype&a
ction=add'>
```

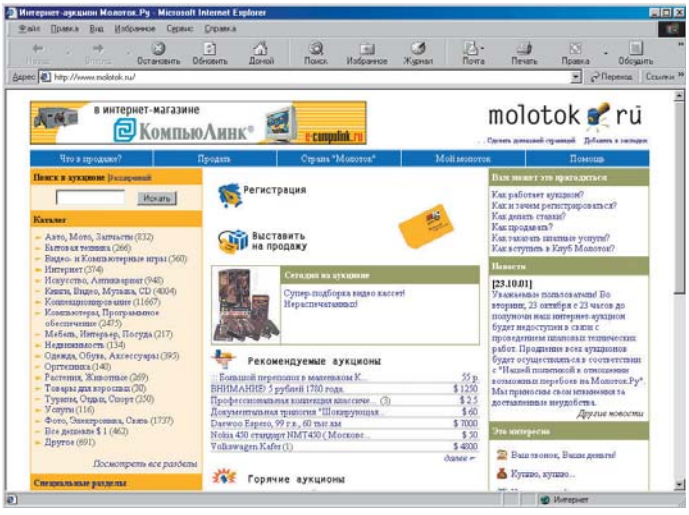
Файл listorder.php отвечает за вывод и работу с корзиной покупок конкретного пользователя, заглянувшего в наш магазинчик. Естественно, что мы должны показать ему его и только его выбранные товары. Вот здесь нам и пригодится переменная $user_session. Ее значение будет служить фильтром для выбираемых записей из таблиц orders и ordered_items.

Номера товарных позиций мы отображать не будем, так как покупателю они совсем неинтересны. Но зато они очень интересны системе, которая по ним понимает, что же из предлагаемого списка он выбрал. Поэтому значения полей id товаров мы поместим в скрытые поля формы в виде массива id[], а введенные пользователем количества поместим в массив qnt[]. Элементов в этих массивах будет столько же, сколько товарных позиций было показано на форме, а из них непустых столько, сколько выбрал покупатель (рис. 2):

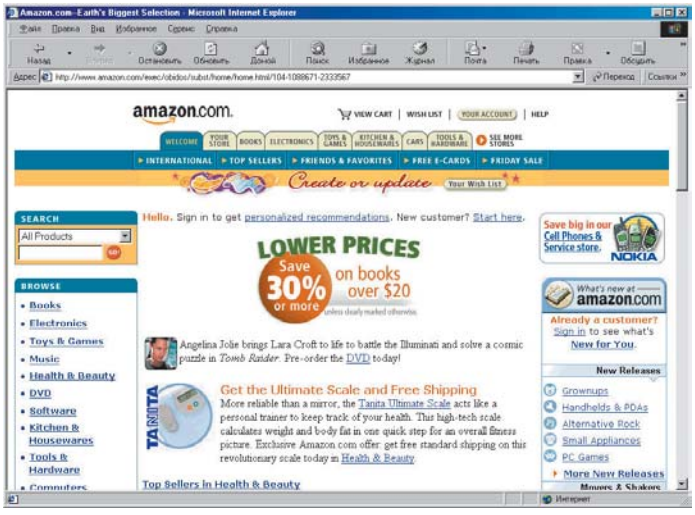
```
function print_item($id,$name,$desc,$price)
{
print "
<tr bgcolor=aqua >
<td width=80%>$name</td>
<td align=center>$price</td>
<td><input type='text' name='qnt[]' value='0'
size='5'></td>
<input type='hidden' name='id[]' value='$id'>
</tr>
<tr><td colspan=3>$desc</td></tr>";
}
```

После нажатия покупателем кнопки SUBMIT будет вызван скрипт listorder.php с параметром action = «add», идентификатором пользовательской сессии и типом выбранной рубрики. В нем уже будут определены массивы $id и $qnt, соответствующие параметрам name в тэгах <input>. В обработчике формы, получившем управление, нужно пройтись по массиву $qnt и обработать элементы, у которых $qnt[$i]>0.

Этим займется класс Order, определенный в файле order.php. Таким образом, скрипт PHP получает массив введенных пользователем величин, не зная заранее ни количества »




▲ Интернет-аукцион Molotok.ru




▲ Универмаг amazon.com

» элементов в нем (оно определяется при обработке), ни какие из них он может использовать (это определяется для каждого элемента). Очень удобная возможность.

## Корзина покупок

В электронном магазине корзина лежит в таблицах базы данных, в то же время она незримо присутствует с покупателем, пока он не отправит заказ продавцу. Покупатель может несколько раз выходить в торговый зал, то есть на страницу listgoods.php, и заказывать новые товары; может увеличить заказанное количество, вспомнив про друзей и приближающийся праздник, может, наоборот, отказаться от покупки какого-то товара — короче, сам управлять своей корзиной. При этом логический блок электронного магазина должен определить, какие товары в данный момент находятся в его корзине, в каком количестве и какова их общая стоимость.

Класс Order должен предоставлять возможность:

- добавить товар в корзину в количестве, введенном покупателем;
- удалить товар из корзины; для простоты мы не рассматриваем изменение количества товара в корзине, но это не составляет большого труда;
- вывести содержимое его корзины;
- подсчитать сумму заказа;
- сформировать по заранее установленной форме письмо с окончательным вариантом заказа и отправить его продавцу, с указанием не только перечня выбранных товаров, но и способа связи с покупателем и адреса доставки ему его покупок.

Для добавления товара в корзину нужно проверить, нет ли там товара с таким же id. Если да, то мы просто прибавляем к нему заказанное количество. В противном случае мы добавляем в заказ новую запись. После этого необходимо пересчитать итоговую сумму. Она будет выведена покупателю при просмотре его корзины. В нашем случае сделать это можно так:

```
function add($id_items,$ord_qnt)
{
for ($i = 0;$i < count($ord_qnt); $i++) {
if ($ord_qnt[$i]>0) {
```

проверим, есть ли такая же позиция в этом заказе,

```
$row = $this->db->select("select count(id) c
from ordered_items
where idOrder=$this->id_order
and idGoods=$id_items[$i]");
if ($row->c == 0)
```

такой нет, добавляем новую,

```
$this->db->sql("insert into ordered_items(i-
dGoods,idOrder,Qnt)
values($id_items[$i],
$this->id_order,$ord_qnt[$i])");
else
```

такая уже есть, добавляем к ней новое количество.

```
$this->db->sql("update ordered_items
set Qnt=Qnt+$ord_qnt[$i]
where idOrder=$this->id_order
and idGoods=$id_items[$i]");
}
}
$this->itog();
}
```

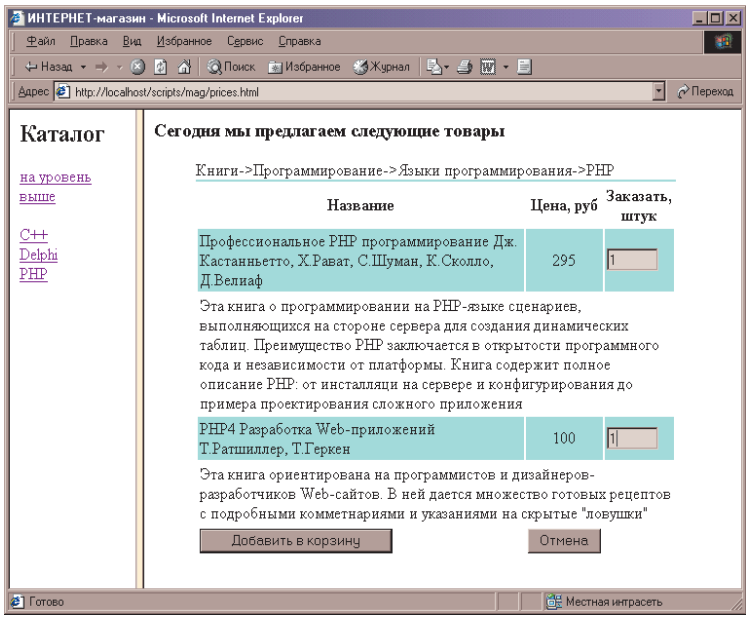
Параметры этого метода — это рассмотренные выше массивы $id и $qnt, полученные из формы выбора товаров. Почему сюда не включена проверка кода сессии?

## Вывод содержимого корзины

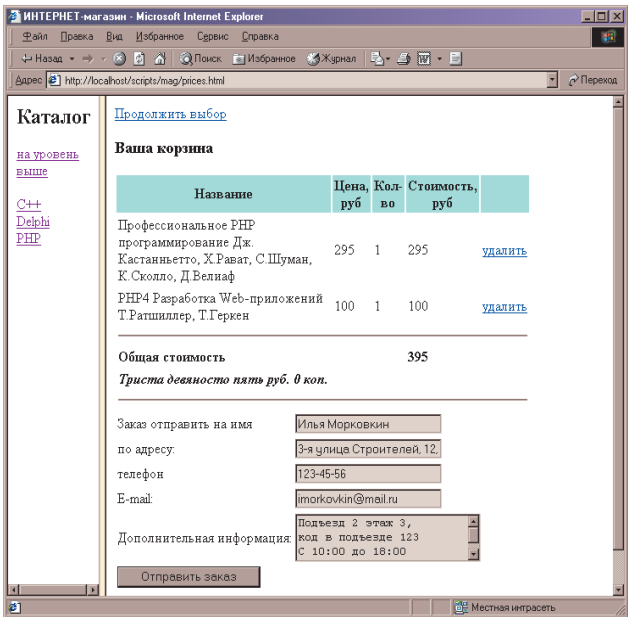
Вывод содержимого корзины осуществляется по тому же принципу, что и вывод остальных компонентов нашего приложения — каталога товаров и списка товарных позиций. Вызываются виртуальные методы печати заголовка таблицы (print_header), каждой строки (print_items) и формы ввода данных о покупателе (print_footer).

Непосредственный вывод, как и в предыдущих классах, обеспечивает его потомок — класс MyOrder, определенный в файле listorder.php, на который передает управление определенная форма заказа товаров.

Особенностью этого скрипта является то, что он обрабатывает два события, переданные ему через параметр action, — добавление и удаление заказанных товаров. Обра- »




▲ Рис. 1. Выбор товаров по каталогу




▲ Рис. 2. Корзина покупок

тить внимание, что запрос на добавление приходит с формы в файле listgoods.php, а на удаление — из самого listorder.php по нажатию пользователем ссылки «удалить». Ссылка эта выводится методом print_items с передачей ему в параметре id удаляемой записи и идентификатора сессии:

```
function
print_items($name,$price,$qnt,$cost,$id)
{
 global $user_session,$idtype;
 print "
<tr>
 <td width='80%'>$name</td>
 <td>$price</td>
 <td>$qnt</td>
 <td>$cost</td>
 <td>
 <a href='listorder.php?
 user_session=$user_session&idtype=$idty-
pe&action=del&id=$id'>
 удалить</a>
</td>
</tr>";
}
```

Кроме того, для записи суммы словами используется функция, определенная в файле bywords.js. Он подключается в заголовке страницы, а из него в методе print_footer печатается вызов функции money_by_words:

```
<script>document.write(money_by_words($t
his->sum))</script>
```

Обработчик события $action выглядит просто как оператор выбора — добавить новый товар или удалить его. После распечатывается текущее содержимое корзины вызовом метода list_order:

```
$order = new MyOrder($user_session);
 switch ($action) {
 case "add" : $order->add($id,$qnt);
 break;
 case "del" : $order->del_item($id); break;
 }
 $order->list_order();
```

Итак, если наш покупатель решил обзавестись парой книг, то корзина с покупками будет выглядеть так, как это показано на рис. 2.

## Спасибо за покупку

После того как пользователь окончательно сформировал свой заказ и отправил его продавцу, требуется сделать три важных действия. Во-первых, дать покупателю подтверждение того, что его заказ принят, и поблагодарить за покупку. Это понятно. Во-вторых, отправить письмо с заказом и внести его данные в базу. В третьих — закрыть заказ.

В таблицу customers заносятся данные о том, кто является покупателем. Позже продавец, обрабатывая заказы и подводя итоги, может предоставить ему скидку или включить лучших покупателей в список рассылки сообщений о новых товарах или в какую-нибудь покупательскую лотерею, то есть проводить маркетинговую работу для привлечения в свой магазин постоянных покупателей. Впрочем, описание маркетинговых ухищрений выходит за рамки данной статьи.

## Завершение работы

Нас интересует сейчас, как функционально завершить работу с заказом. Этим займется скрипт confirm.php. Мы закрываем заказ, назначая сеансу новый идентификатор. Теперь со старым идентификатором сессии уже не будет выполняться никаких операций, кроме последней — отправки письма продавцу. Для этой цели мы его и сохранили в переменной $sess. Если покупатель захочет вернуться в магазин и докупить что-нибудь еще, он будет формировать новый заказ с пустой корзины.

Для отправки заказа воспользуемся экземпляром класса Order:

```
$order = new Order($sess);
```

Отправим письмо с заказом.

```
 $order-
>send(strip_tags($name),strip_tags($ad-
dress),strip_tags($phone),
 strip_tags($email),strip_tags($additional),
 $letter_header,$letter_detail,$letter_footer);
```

Мы создаем экземпляр класса Order и методом send отправляем продавцу письмо с заказом, в которое подставлены значения из

**Привязка к сессии и общая сумма**

# Во избежание недоразумений

Таблица позиций заказа ссылается на номер заказа, а он, в свою очередь, на запись о заказе с соответствующим значением поля idSession. В конструкторе класса Order мы и проверяем наличие заказа с таким идентификатором сессии и запоминаем его номер в свойстве $this->id_order.

```
 function Order($user_session) {
 global $host,$username,$password,
$dbname;
 $this->session = $user_session;
 $this->db = new
DB($host,$username,$password,$dbname);
```

Добавим заказ, если его еще не было

```
 $row = $this->db->select("select id from
orders where idSession='$this->session'");
 if (empty($row->id)) {
```

таких нет, добавляем новый заказ,

```
 $this->db->sql("insert into orders
(idSession,date,sum)
values('$this->session',Now(),0)");
 $this->id_order = $this->db->insert_id();
 }  else $this->id_order = $row->id;  }
```

Таким образом посредством свойства id_order мы осуществили привязку заказа к текущей сессии конкретного покупателя.
Метод подсчета итога определяет суммарную стоимость товаров и пишет ее в таблицу orders, устанавливая ее значение в свойство sum. Значение sum можно использовать при выводе содержимого корзины.

```
 function itog() {
 $row = $this->db->select("select
ifnull(sum(price*Qnt),0) S from ordered_i-
tems I,goods G,orders O where O.idSessi-
on='$this->session' and I.idGoods=G.id and
O.id=I.idOrder");
 $this->db->sql("update orders set
sum=$row->S where id=$this->id_order");
 $this->sum = $row->S;
 return $this->sum; }
```

Этот метод необходимо вызывать после каждой модификации корзины во избежание досадных недоразумений.

формы, заполненной покупателем — $name, $address, $phone, $email, $additional. Переменные $letter_header, $letter_detail и $letter_footer представляют собой HTML-шаблоны для формирования текста письма. В них подставляются значения, и полученный текст помещается в тело письма, которое и получит продавец. Здесь следует учитывать вероятность, что злоумышленник может вставить в текст нежелательные тэги, после чего они письмом отправятся к продавцу. Поможет убирать все тэги из строки библиотечная функция strip_tags.

После отправки письма продавцу с перечислением товаров и количества, у покупателя выводится страница с подтверждением отправки заказа (рис. 3).

## Метод send

Рассмотрим использованный метод send класса Orders. Из тех же соображений разделения функциональной части и части вывода, необходимые шаблоны для построения письма передаются этому методу в параметр. Если продавец захочет что-либо изменить в формате письма, то ему незачем лезть в класс Orders, который непосредственно отправляет ему заказ. Также в параметр передаются данные о покупателе, полученные из формы. Метод send() делает следующее:

```
function
send($Name,$Address,$Phone,$Email,$Additional,
 $lheader,$ldetail,$lfooter)
 {
 global $order_email;
```

добавим пользователя,

```
 $this->db->sql("insert into customers
 (name,address,phone,email,additions,idOrder)
 values('$Name','$Address','$Phone','$Email','$Additional',
 $this->id_order)");
```

Сформируем и отправим письмо,

```
 $mess = sprintf($lheader,$this->id_order,date('d.m.Y'),$Name,
 $Phone,$Email,$Email);
 $row = $this->db->select("
 select G.name,I.Qnt,G.price,I.Qnt*G.price as cost,I.id
 from orders O, ordered_items I, goods G
 where O.id=I.idOrder and G.id=I.idGoods
 and O.id='$this->id_order'");
 while (!$this->db->eof()) {
  $mess .= sprintf($ldetail,$row->name,$row->price,$row->Qnt,$row->cost);
  $row = $this->db->next();
 }
 $this->itog();
 $mess .= sprintf($lfooter,$this->sum,$Address,$Additional);
 mail($order_email,"Заказ $this->id_order",$mess,
 'Content-Type: text/html; charset=win-1251\n');
 }
}
```

Вначале добавляется запись о покупателе. Затем последовательно формируется сообщение, так же как и при выводе на web-страницу, только вместо оператора вывода, сформированная строка «приклеивается» к телу сообщения $mess. Думается, программисты на C легко узнают здесь знакомую функцию sprintf. В PHP она работает точно также, формируют строку из параметров по шаблону. Шаблоны мы определили в файле confirm.php и передали в метод send в качестве параметров $lheader, $ldetail, $lfooter. Для тех, кто не работал на C, поясню, что в шаблоне символ %s означает подстановку на это место параметра строкового типа, а %d означает подстановку на это место целого числа из параметра.

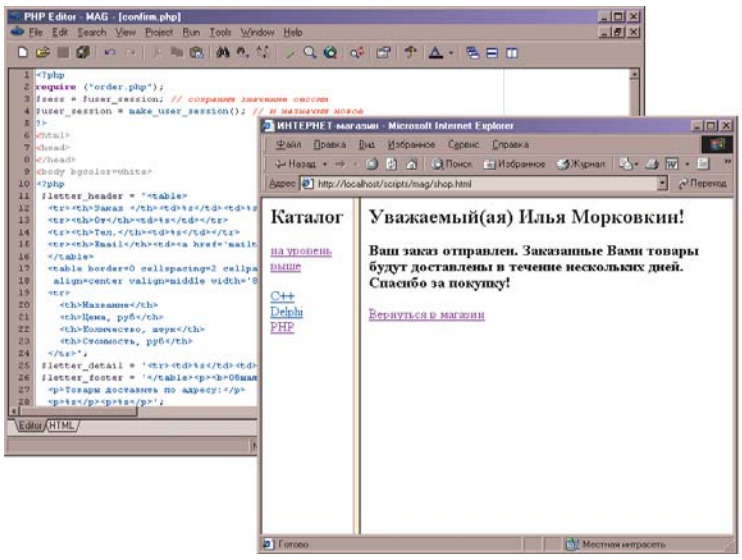
Отправляет сообщение функция mail. Параметры у нее такие: адрес, тема, тело сообщения в виде строки и заголовок MIME. Продавец получит письмо с заказом в формате, показанном на рис. 4.

## Заключение

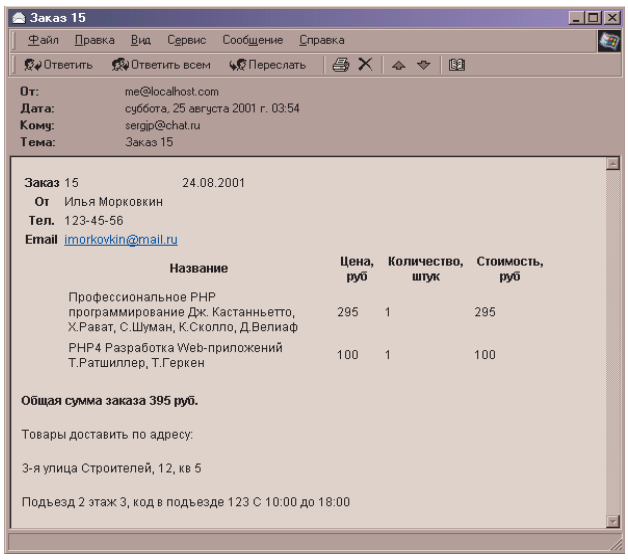
Мы рассмотрели принципы построения простейшего онлайнового магазина в Интернете. Программное обеспечение реальных магазинов, торгующих через Интернет, таких как www.mistral.ru или www.amazon.com, решает значительно более сложные задачи и содержит многие тысячи строк кода.

Исходные коды, приведенные в данной статье и на Chip CD, не претендуют на звание полностью готового приложения, но эти материалы вполне могут служить основой для построения коммерческой системы продажи товаров через Интернет.

■ ■ ■ Сергей Бабичев




▲ Рис. 3. Подтверждение приема заказа




▲ Рис. 4. Письмо с заказом