

Виктор Петин

Сайт на АЈАХ под ключ

**Готовое решение
для интернет-магазина**

Санкт-Петербург

«БХВ-Петербург»

2011

УДК 681.3.06
ББК 32.973.26-018.2
П29

Петин В. А.

П29 Сайт на AJAX под ключ. Готовое решение для интернет-магазина. — СПб.: БХВ-Петербург, 2011. — 432 с.: ил. + CD-ROM — (Профессиональное программирование)

ISBN 978-5-9775-0629-8

Описана разработка высоконтерактивных Web-сайтов, основанных на передовой технологии AJAX, работающих без перезагрузки страниц и обладающих функциональностью настольных приложений. Обучение построено на сквозном примере создания с нуля готового решения: интернет-магазина цифровых товаров, а также системы его администрирования. При этом использован язык PHP, фреймворки хайах и jQuery, шаблонизатор Smarty и другие популярные технологии динамического формирования контента. Разработанный сайт создан полностью по технологии AJAX и готов к размещению в сети.

Прилагаемый компакт-диск содержит исходные коды описанного в книге интернет-магазина, а также бесплатные программы для создания и отладки сайтов на локальной машине.

Для Web-разработчиков

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	Екатерина Кондукова
Зам. главного редактора	Евгений Рыбаков
Зав. редакцией	Григорий Добин
Редактор	Леонид Кочин
Компьютерная верстка	Наталья Караваевой
Корректор	Виктория Пиотровская
Дизайн серии	Инны Тачиной
Оформление обложки	Елены Беляевой
Зав. производством	Николай Тверских

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 26.10.10.

формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 34,83.

Тираж 1000 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диагпозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Введение.....	1
Для кого и о чём эта книга	1
Структура книги.....	2
Благодарности	3

ЧАСТЬ I. ИНСТРУМЕНТЫ И ТЕХНОЛОГИИ WEB-ПРОГРАММИРОВАНИЯ 5

Глава 1. Инструменты создания Web-страниц.....	7
1.1. HTML и CSS	7
1.1.1. Теговая модель	7
1.1.2. Элементы HTML	8
1.1.3. Классификация элементов HTML	8
1.1.4. Атрибуты тегов.....	9
1.1.5. Листы стилей CSS	9
1.1.5.1. Определение встроенного стиля	9
1.1.5.2. Формирование листа стилей.....	9
1.1.5.3. Внутренние листы стилей.....	10
1.1.5.4. Внешние листы стилей.....	10
1.2. Язык сценариев JavaScript.....	11
1.2.1. Встраивание сценария JavaScript в документ	11
1.2.2. Обработка событий в JavaScript.....	12
1.3. Динамический HTML	13
1.4. PHP — серверный язык программирования.....	16
1.5. СУБД MySQL	17
1.5.1. Типы данных.....	17
1.5.1.1. Целые числа	17
1.5.1.2. Дробные числа	18
1.5.1.3. Строки.....	18
1.5.1.4. Бинарные данные	19
1.5.1.5. Дата и время	19
1.5.2. Таблицы MySQL.....	20

1.5.3. Структурированный язык запросов SQL	20
1.5.4. Функции PHP для работы с MySQL	22
1.5.4.1. <i>mysql_connect</i>	22
1.5.4.2. <i>mysql_close</i>	22
1.5.4.3. <i>mysql_select_db</i>	22
1.5.4.4. <i>mysql_query</i>	23
1.5.4.5. <i>mysql_fetch_row</i>	23
1.5.4.6. <i>mysql_fetch_assoc</i>	23
1.5.4.7. <i>mysql_fetch_array</i>	24
1.5.4.8. <i>mysql_result</i>	24
1.5.4.9. <i>mysql_num_rows</i>	24
1.5.4.10. <i>mysql_insert_id</i>	24
1.5.5. Работа с phpMyAdmin	25
1.5.5.1. Запуск phpMyAdmin из Денвера	25
1.5.5.2. Создание базы данных	26
1.5.5.3. Создание таблицы базы данных	26
1.5.5.4. Заполнение таблиц базы данных	28
1.5.5.5. Экспорт-импорт баз данных	29
1.6. Программная оболочка Денвер	31
1.6.1. Что такое Денвер?	31
1.6.2. Получение дистрибутива и расширений Денвера	32
1.6.3. Установка Денвера	35
1.6.4. Размещаем сайт на локальном компьютере	39
Глава 2. Технология AJAX.....	44
2.1. Что такое AJAX	44
2.1.1. Обмен данными между клиентом и сервером	44
2.1.2. Свойства и методы объекта XMLHttpRequest	45
2.1.3. Запрос к серверу и обработка ответа	46
2.1.4. Варианты ответа от сервера	47
2.2. Фреймворк хаях	48
2.2.1. Как работает хаях	48
2.2.2. Возможности хаях	48
2.2.3. Подключение хаях	49
2.2.4. Методы объекта <i>xajaxResponse</i>	51
2.2.4.1. Метод <i>assign</i>	51
2.2.4.2. Метод <i>append</i>	52
2.2.4.3. Метод <i>prepend</i>	52
2.2.4.4. Метод <i>replace</i>	52
2.2.4.5. Метод <i>remove</i>	53
2.2.4.6. Метод <i>create</i>	53
2.2.4.7. Метод <i>insert</i>	53
2.2.4.8. Метод <i>insertAfter</i>	53

2.2.4.9. Метод <i>clear</i>	54
2.2.4.10. Метод <i>createInput</i>	54
2.2.4.11. Метод <i>insertInput</i>	54
2.2.4.12. Метод <i>insertInputAfter</i>	54
2.2.4.13. Метод <i>removeHandler</i>	55
2.2.4.14. Метод <i>includeScript</i>	55
2.2.4.15. Метод <i>script</i>	55
2.2.4.16. Метод <i>addEvent</i>	56
2.2.4.17. Метод <i>call</i>	56
2.2.4.18. Метод <i>alert</i>	56
2.2.4.19. Метод <i>redirect</i>	56
2.2.5. Сайт — тренировочный стенд для изучения хаах	57
2.2.6. Глобальные переменные хаах	61
2.2.6.1. Глобальные константы	61
2.2.6.2. Методы объекта хаах	61
2.3. Примеры использования хаах	65
2.3.1. Форма регистрации с проверкой правильности заполнения полей "на лету"	65
2.3.2. Динамически подгружаемые select-элементы	70
2.3.3. Многоуровневый неоднородный каталог	79
2.3.4. Динамическое управление числом полей формы	84
2.4. Фреймворк jQuery	92
2.4.1. Возможности jQuery	92
2.4.2. Использование jQuery	93
2.4.2.1. Функция \$	93
2.4.2.2. Селекторы	94
2.4.2.3. Методы jQuery	97
2.4.2.4. Обработка событий в jQuery	98
2.4.2.5. Эффекты в jQuery	100
2.4.3. PHP и jQuery	101
2.4.3.1. Динамическая подгрузка jQuery и плагина Carousel	101
2.4.3.2. Совместное использование jQuery UI, виджетов Tabs и Accordion	104
2.4.3.3. Галерея товаров с формой заказа	113
2.5. Хаах и Smarty	127
2.5.1. Что такое Smarty	127
2.5.2. Установка Smarty	128
2.5.3. Синтаксис шаблонов Smarty	130
2.5.4. Методы класса Smarty	131
2.5.4.1. Метод <i>assign</i>	131
2.5.4.2. Метод <i>display</i>	132
2.5.4.3. Метод <i>fetch</i>	132
2.5.5. Использование хаах и Smarty	132

ЧАСТЬ II. ПРОЕКТ ИНТЕРНЕТ-МАГАЗИНА	139
Глава 3. Проектирование сайта	141
3.1. Структура и функции сайта	141
3.1.1. Необходимый функционал сайта (интернет-магазина цифровых товаров)	141
3.1.2. Структура корневого каталога сайта	143
3.1.3. Особенности создания сайта без перезагрузки страницы	143
3.1.4. Проектирование базы данных	145
3.2. Типы пользователей. Вход в профиль	155
3.2.1. Типы пользователей	155
3.2.2. Вход в профиль	156
3.2.3. Использование переменных <i>SESSION</i> и <i>cookies</i>	160
3.2.3.1. Переменные <i>session</i>	161
3.2.3.2. Переменные <i>cookie</i>	162
3.2.4. Логика вызова программ при выборе пункта меню	164
3.2.5. Набор подпрограмм модулей для разных пользователей	171
3.3. Регистрация	180
3.3.1. "Теневая" регистрация незарегистрированных пользователей	180
3.3.2. Регистрация пользователей	182
3.4. Оплата SMS через сервис aLaggregator	190
3.5. Блок "Товары"	195
3.5.1. Список категорий товаров неограниченной вложенности	195
3.5.2. Вывод списка товаров постранично	199
3.5.3. Динамический "рессайзер" картинок	204
3.5.4. Программирование навигатора страниц	206
3.5.5. Вывод пути к категории товаров	208
3.5.6. Поиск товаров и вывод постранично	210
3.5.7. Просмотр товара подробно	217
3.5.8. Специальные акции (товары по акции)	220
3.6. Корзина	222
3.6.1. Добавление товаров в корзину	222
3.6.2. Корзина подробно	227
3.6.3. Редактирование корзины	230
3.6.3.1. Изменение количества товара	230
3.6.3.2. Удаление товара из корзины	231
3.6.4. Оформление заказа	234
3.7. Оплата заказа	237
3.7.1. Оплата Webmoney	237
3.7.2. Организация приема платежей Webmoney	242
3.7.3. Платежный интегратор ONPAY	245
3.7.3.1. Варианты приема электронных платежей	246
3.7.3.2. Настройка параметров магазина	247
3.7.3.3. ONPAY Merchant API	249

3.7.4. Подключение приема платежей в автоматическом режиме через ONPAY Merchant API	257
3.8. Блок "Заказы"	263
3.8.1. Просмотр заказов пользователя	263
3.8.2. Поиск заказов пользователя по фильтру	267
3.8.3. Редактирование заказа	273
3.8.4. Просмотр заказа	281
3.8.5. Удаление заказа	284
3.8.6. Оплата заказа. Формирование ссылок для скачивания	285
3.8.7. Регулирование доступа к файлам скачивания с использованием файла .htaccess	287
3.8.8. Получение товара	288
3.9. Блок мгновенных сообщений на сайте	292
3.9.1. Вывод мгновенных сообщений	292
3.9.2. Переход по ссылке мгновенных сообщений	294
3.9.3. Формирование мгновенных сообщений	296
3.10. Переписка на сайте (внутренняя почта)	299
3.10.1. Просмотр сообщений пользователя списком	299
3.10.2. Просмотр сообщения	306
3.10.3. Удаление сообщения	308
3.10.4. Создание сообщения	310
Глава 4. Программирование панели администратора	315
4.1. Вход администратора	315
4.2. Управление товарами	316
4.2.1. Добавление нового товара	317
4.2.2. Редактирование товара	332
4.2.3. Удаление товара	339
4.2.4. Скрытие товара, открытие товара	340
4.3. Управление категориями товаров	343
4.3.1. Добавление категорий товаров	347
4.3.2. Редактирование категорий товаров	350
4.3.3. Удаление категорий товаров	353
4.4. Управление заказами	356
4.4.1. Просмотр заказов пользователей	356
4.4.2. Просмотр заказов пользователей по фильтру	360
4.4.3. Просмотр заказа	367
4.4.4. Редактирование заказа	370
4.4.5. Удаление заказа	376
4.4.6. Оплата заказа администратором	377
4.5. Операции с профилями пользователей	379
4.5.1. Просмотр всех пользователей	379
4.5.2. Просмотр пользователей по фильтру	383

4.5.3. Просмотр профиля пользователя.....	389
4.5.4. Редактирование профиля пользователя	391
4.5.5. Блокировка пользователя	396
4.6. Обратная связь.....	397
4.6.1. Обратная связь по e-mail.....	397
4.6.2. Обратная связь по ICQ	400
Заключение	404
ПРИЛОЖЕНИЯ	405
Приложение 1. Свойства стилей CSS.....	407
Приложение 2. Описание компакт-диска	421
Предметный указатель	423

Введение

Для кого и о чём эта книга

Предлагаемая книга ориентирована на читателей, владеющих языком разметки HTML, имеющих общее представление о языке JavaScript и обладающих навыками программирования сайтов на языке PHP.

Имея некоторый опыт разработки сайтов, вы наверняка хотели бы овладеть новыми передовыми технологиями программирования, одной из которых является AJAX, чтобы с их помощью создавать Web-приложения, удовлетворяющие самым современным требованиям. Посмотрите Gmail, Google Maps, Google Suggest и десятки других Web-проектов, которые предлагают новый уровень интерактивного интерфейса. Современные Web-приложения могут быть разработаны с расширенным пользовательским интерфейсом и функциями, которые раньше были привилегией профессиональных приложений. AJAX позволяет создавать более интерактивные, быстroredействующие и гибкие решения для Интернета. И это лишь первый шаг на пути к приложениям с более широкими возможностями в Интернете.

Вы думаете, такая задача непосильна для вас? Докажу обратное. В этой книге мы вместе создадим готовое решение — интернет-магазин цифровых товаров, реализованный полностью по технологии AJAX, без единой перезагрузки страницы.

Может быть, вы думаете, что это слишком трудно, т. к. недостаточно хорошо знаете JavaScript? На самом деле создание сайта потребует самых минимальных знаний JavaScript, поскольку весь функционал написан на PHP.

Мы вместе с вами создадим сайт, готовый к размещению в сети Интернет, исходные коды которого представлены на прилагаемом к книге компакт-диске. Архитектура сайта позволит быстро изменить его или переделать под другое решение.

Кроме того, в книге вы найдете множество примеров. Примеры представляют собой законченные решения, пригодные для использования в ваших проектах.

Структура книги

Книга состоит из двух частей, включает введение, четыре главы, заключение и два приложения.

Первая часть содержит описание инструментов и технологий, применяемых при разработке Web-сайтов.

В главе 1 "Инструменты создания Web-страниц" обзорно рассмотрим инструменты для создания Web-страниц: язык разметки HTML, динамический HTML (DHTML) и язык сценариев JavaScript, СУБД MySQL. Рассмотрим также структурированный язык запросов SQL, Web-интерфейс к MySQL — phpAdmin, а также функции PHP для работы с MySQL. Здесь же опишем установку и настройку программной оболочки Денвер — набора программ для отладки сайтов на локальной Windows-машине.

Глава 2 "Технология AJAX" посвящена рассмотрению технологии AJAX и популярных фреймворков хаях и jQuery. В главе приведены примеры изменения динамического содержания страницы по технологии AJAX с использованием фреймворка хаях, а также реализация возможностей библиотеки jQuery в PHP. Все предлагаемые примеры — рубрикатор населенных пунктов России с подгрузкой содержимого "на лету", форма заказов с динамическим изменением количества полей, многоуровневый каталог, красочный мини-магазин с галереей видов каждого товара и формой заказа, заготовка под портал — с минимальными переделками подойдут для ваших проектов. В последнем разделе "Хаях и Smarty" мы на примерах рассмотрим применение шаблонизатора Smarty при создании сайтов без перезагрузки страницы: формирование главной страницы и динамическую подгрузку результатов запросов к серверу.

Вторая часть книги целиком посвящена разработке конкретного проекта — интернет-магазина.

В главе 3 "Проектирование сайта" рассмотрим особенности реализации сайтов без перезагрузки страницы, определим необходимый функционал разрабатываемого сайта — интернет-магазина, спроектируем для него структуру базы данных и запрограммируем пользовательский интерфейс.

Глава 4 "Программирование панели администратора" посвящена разработке подпрограмм панели администратора.

В *приложении 1* приведены свойства стилей. В *приложении 2* дано описание компакт-диска, прилагаемого к книге.

Благодарности

Хочу поблагодарить родных и близких, которые с пониманием относились к потраченному на книгу (за счет общения с ними) времени.

Большая благодарность издательству "БХВ-Петербург", где поверили в необходимость данной книги, и всем сотрудникам издательства, которые помогали мне в создании книги.

Хочу поблагодарить также всех читателей, которые купят эту книгу, я делал все, чтобы она была интересной и полезной, надеюсь, что так оно и есть.

Если возникнут вопросы или пожелания по данной книге, то вы всегда сможете со мной связаться по электронной почте victorguni@km.ru, kmvnews@bk.ru или оставить сообщение в блоге <http://goodtovars.ru/blog>, где рассматриваются вопросы создания сайтов без перезагрузки страницы.



ЧАСТЬ I

**Инструменты и технологии
Web-программирования**



Глава 1

Инструменты создания Web-страниц

Основа программирования документов для Web — язык разметки HTML — позволяет создавать только статические страницы, обновляемые с сервера. В отличие от обычного HTML, динамический HTML (DHTML) обеспечивает взаимодействие Web-документов с пользователем и дает возможность изменять документ на компьютере клиента без обращения на сервер. Инструментом для манипулирования страницами на компьютере клиента служат языки сценариев JavaScript и VBScript, из которых в настоящее время наиболее популярен JavaScript. Однако для создания по-настоящему динамических Web-приложений (взаимодействие с посетителями, получение от них информации, настройка страниц под конкретного пользователя и т. д.) необходимо взаимодействие страниц с сервером. Было создано несколько серверных языков для написания сценариев на стороне сервера и формирования динамических страниц. PHP — один из самых успешных таких языков — быстро нашел свое применение и приобрел большую популярность. При разработке Web-приложений нам понадобится сервер баз данных. В этой главе рассмотрим один из наиболее подходящих для нас — MySQL. Существенно облегчит вашу работу отладочный пакет Денвер, описанный в последнем разделе.

1.1. HTML и CSS

Всемирная "паутина" основана на языке гипертекстовой разметки HTML. HTML не вполне обычный язык: он не относится к языкам высокого уровня. Это язык разметки, и код, написанный на нем, исполняется на компьютере клиента в приложении Web-браузера. Web-страница, загружаемая в браузер, представляет собой HTML-файл. Для просмотра HTML-кода документа щелкните правой кнопкой мыши в окне документа и в появившемся контекстном меню выберите в зависимости от браузера команду **Просмотр HTML-кода** или **Исходный код страницы**.

1.1.1. Теговая модель

Разметка HTML-документов выполняется с помощью тегов, которые записываются с соблюдением определенных правил. Теговая модель предполагает разбиение документа на отдельные фрагменты, которые заключаются в теги или начинаются тегом. Все теги начинаются с открывающейся угловой скобки <, за которой следует

текст, определяющий содержание тега. Оканчивается тег закрывающейся угловой скобкой >. Содержанием тега может быть просто его имя, либо имя и набор атрибутов тега.

Большинство тегов парные, для каждого начального тега <имя> есть конечный тег </имя>, например:

```
<TABLE> . . . </TABLE>  
<FORM> . . . </FORM>
```

Многоточие означает, что между начальным и конечным тегами может находиться текст или другие теги.

Кроме парных, существуют одиночные теги, в которых имеется только открывающий тег. В соответствии с инструкциями одиночных тегов браузер выполняет определенные действия, например:

- <P> — формирование нового абзаца;
- <HR> — вставка горизонтальной линии;
- — вставка изображения.

1.1.2. Элементы HTML

Документ HTML включает в себя элементы, которые представляют абзацы, заголовки, списки, таблицы, гиперссылки, рисунки и пр. Весь документ можно рассматривать как совокупность определенных элементов. Элемент — это пара тегов и символьные данные (код или текст), заключенные между ними. Иначе говоря, элемент состоит из начального тега, содержащего и конечного тега. В некоторых элементах конечный тег может быть опущен (в случае одиночных тегов). Элементы HTML не чувствительны к регистру символов, т. е. браузер одинаково воспринимает теги <Table>, <TABLE>, <table>. Список элементов HTML утвержден спецификацией HTML 4.01. Если браузер находит незнакомый элемент, он его просто игнорирует.

1.1.3. Классификация элементов HTML

Все элементы, предусмотренные в HTML, можно условно разбить на несколько категорий:

- структурные — обязательны для документа, соответствующего стандарту HTML (например, HTML, HEAD, BODY, TITLE);
- блоковые — предназначены для форматирования целых текстовых блоков (например, DIV, H1, H2, PRE);
- текстовые — создают разметку текста (EM, STRONG) и разметку шрифта (I, B, U);
- специальные — элементы пустой строки (BR, HR), ссылка A, внедренные элементы (IMG, EMBED, OBJECT), элементы формы (INPUT, SELECT, TEXTAREA), элементы таблицы (TABLE, TR, TD) и др.

1.1.4. Атрибуты тегов

Часто теги, помимо имени, содержат дополнительные элементы, которые называют атрибутами:

```
<H1 id="zagolovok" color="red" ></H1>
```

Атрибут записывается после имени тега перед закрывающейся скобкой `>` и состоит из пары имя атрибута=значение. В теге может быть несколько атрибутов. Атрибуты отделяют друг от друга пробелами, очередность записи атрибутов в теге не имеет значения. Атрибуты записываются в начальных тегах и отсутствуют в конечных. Одно из главных назначений атрибутов — управлять видом элемента на странице положением, цветом.

1.1.5. Листы стилей CSS

Каскадные листы (таблицы) стилей (CSS) — это язык, используемый в документах HTML для определения способа представления содержимого. Представление задается с помощью стилей, помещаемых непосредственно в элементы HTML, заголовок HTML-документа или отдельные таблицы стилей. В листах стилей значение свойства присоединяется к стилю при помощи двоеточия:

```
background-color : green  
font-size: 12 pt
```

1.1.5.1. Определение встроенного стиля

Простейший способ задания стиля элемента HTML с помощью атрибута `style`:

```
<div style="font-size:12pt;color:green;background-color:yellow">
```

Введение стиля мало чем отличается от форматирования средствами HTML. В то же время встроенный стиль — простейший способ задания стилевых свойств, который можно оперативно применить к отдельным элементам в процессе создания документа. При этом нужно понимать, что встроенные стили нарушают основную концепцию CSS, заключающуюся в том, что форматирование документа должно быть отделено от содержания.

1.1.5.2. Формирование листа стилей

Стилевые свойства вводят с помощью определения стиля, которое принято обозначать фигурными скобками:

```
{ font-family:Arial; background-color:yellow }  
{visibility:hidden}
```

Назначение стиля тому или иному элементу состоит в установлении связи:

```
span {color:red;font-style:italic}  
div.p {text-size:12 px; margin-left: 10 px}  
#div1 {border-color:green}
```

Элемент, к которому относится определенный стиль, называется селектором. Селекторы, записанные прописными буквами, обозначают классы, а селекторы, начинающиеся со знака `#`, отвечают уникальным идентификаторам элементов.

1.1.5.3. Внутренние листы стилей

Встроенные стили имеют большой недостаток — они не позволяют отделить средства форматирования документа от его содержания. Кроме того, объявления встроенного стиля приходится повторять для каждого форматируемого элемента на протяжении всего документа. От этих недостатков свободен другой способ введения стилей — размещение листа стилей в заголовочной части документов. Согласно этому способу, называемому заголовочным стилем, можно единым образом управлять содержимым всего документа. Для изменения отображения одинаково оформленных элементов достаточно один раз изменить соответствующие свойства в листе стилей. Встроенные и заголовочные стили относятся к внутренним листам стилей. Для введения заголовочного стиля в заголовочную часть HTML-документа вставляется специальный контейнер `<style></style>`:

```
<style type="text/css">
<!-- описание листа стилей -- &gt;
&lt;/style&gt;</pre>
```

Пример

```
<head>
<style="text/css">
h4.style1 {color:red}
#style2 {color:green;background-color:yellow}
</style>
</head>
```

Сопоставление правил CSS с конкретными элементами документа выполняется с помощью атрибутов `class` и `id`:

```
<body>
<h4 class="style1">Заголовок 1</h4>
<div id="style2">Текст 1</div>
</body>
```

1.1.5.4. Внешние листы стилей

Внешние листы стилей записываются в отдельных файлах и применяются для оформления набора HTML-документов. Использование внешних листов стилей позволяет единым образом оформлять множество Web-страниц и даже сайтов. Удобство внешних стилей заключается также и в том, что можно изменять стили, не затрагивая содержания документов. Описание стилей хранится в отдельном файле, который имеет расширение `css`. Содержательная часть CSS-файла состоит только из листа стилей. Основным инструментом подключения к HTML-документу

внешних листов стилей является одиночный тег `<LINK>`, который располагается в заголовочной части `<HEAD>`:

```
<HEAD>
<LINK type="text/css" href="http://www.my_site.ru/css/
site.css" rel="stylesheet" ">
</HEAD>
```

где:

- `type="text/css"` — указывает браузеру, что применяется текст формата CSS;
- `href` — задает URL файла внешнего листа стилей;
- `rel="stylesheet"` — указывает на то, что элемент `LINK` устанавливает связь с внешним листом стилей.

1.2. Язык сценариев JavaScript

JavaScript — это один из основных языков разработки Web-страниц, который поддерживают все популярные браузеры. Для просмотра Web-страниц, содержащих инструкции JavaScript, пользователю не нужно устанавливать дополнительное программное обеспечение. Язык JavaScript разработан компанией Netscape Communications и является языком сценариев. Этот язык призван был расширить возможности HTML по переработке информации из форм и добавлению динамики на Web-страницы. JavaScript вначале был задуман как клиентский язык, предназначенный для работы на компьютере клиента-пользователя. Идея создания JavaScript заключалась именно в возможности размещения на Web-страницах исполняемого содержимого, благодаря чему можно было бы выйти за рамки статического HTML, обеспечить взаимодействие с пользователем, управление браузером и т. д. Однако по мере своего развития JavaScript вышел за рамки отдельно взятого браузера и стал выполнять также функции серверной части.

1.2.1. Встраивание сценария JavaScript в документ

На стороне клиента сценарий содержится в HTML-файле. Код сценария заключен между тегами `<SCRIPT>` и `</SCRIPT>`, которые могут быть размещены в любом месте HTML-документа вслед за тегами `<HEAD>` и `<BODY>`. В документе может быть несколько сценариев, ограниченных тегами `<SCRIPT>` и `</SCRIPT>`, причем эти фрагменты не должны перекрываться. Сценарии будут исполняться в порядке их расположения в документе. Функции исполняются при обращении к ним обработчиков событий или при вызове из других функций. В теге `<SCRIPT>` обязательно нужно указывать язык сценария:

```
<SCRIPT language="JavaScript">
<!-- Операторы языка JavaScript -- &gt;
&lt;/SCRIPT&gt;</pre>
```

Возможны следующие случаи размещения сценария в HTML-документе:

- в теле программы (между тегами <BODY>), в этом случае сценарий исполняется при загрузке страницы в браузер;
- в заголовке документа (между тегами <HEAD>), в этом случае сценарий не исполняется сразу при загрузке, а может использоваться как функция другими сценариями;
- между тегами HTML (между угловыми скобками < ...>), при этом сценарий является обработчиком событий, для его записи не требуются теги <SCRIPT>;
- в отдельном файле. Язык JavaScript допускает создание собственных файлов с расширением js, в которых размещаются сценарии:

```
<SCRIPT language="JavaScript" src="js/jquery-1.4.2.js">
```

1.2.2. Обработка событий в JavaScript

Очень важное место в программировании Web-страниц занимают события. События генерируются в результате действий пользователя (щелчков мыши, нажатия клавиш и пр.). Разрабатывая Web-страницы, можно составить сценарий таким образом, что страница будет реагировать на некоторые из событий. Это делается с помощью специальных программ, которые называются обработчиками событий. Программы обработчиков событий представляют собой фрагменты кода и обычно оформляются в виде функций. Обработчик событий, написанный на JavaScript, вводится в сценарий просто — буквально одной строкой, например:

```
<input name=input1 type=text onclick="alert(this.value) ; ">
<input name=input2 type=text onchange="function1(this.value)">
```

В табл. 1.1 приведены события и элементы HTML, в которых эти события могут происходить.

Таблица 1.1. События и объекты

Обработчик события	Поддерживающие HTML-элементы	Описание
onAbort	IMG	Прерывание загрузки изображения
onBlur	A, AREA, BUTTON, INPUT, LABEL, SELECT, TEXTAREA	Потеря текущим элементом фокуса, т. е. переход к другому элементу. Возникает при щелчке мышью вне элемента либо при нажатии клавиши табуляции
onChange	INPUT, SELECT, TEXTAREA	Изменение значений элементов формы. Возникает после потерей элементом фокуса, т. е. после события blur
onClick	Практически все HTML-элементы	Одинарный щелчок (нажата и отпущена кнопка мыши)
onDblClick	То же	Двойной щелчок

Таблица 1.1 (окончание)

Обработчик события	Поддерживающие HTML-элементы	Описание
onError	IMG, WINDOW	Возникновение ошибки выполнения сценария
onFocus	A, AREA, BUTTON, INPUT, LABEL, SELECT, TEXTAREA	Получение элементом фокуса (щелчок мышью на элементе или очередное нажатие клавиши табуляции)
onKeyDown	Практически все HTML-элементы	Нажата клавиша на клавиатуре
onKeyPress	То же	Нажата и отпущена клавиша на клавиатуре
onKeyUp	То же	Отпущена клавиша на клавиатуре
onLoad	BODY, FRAMESET	Закончена загрузка документа
onMouseDown	Практически все HTML-элементы	Нажата кнопка мыши в пределах текущего элемента
onMouseMove	То же	Перемещение курсора мыши в пределах текущего элемента
onMouseOut	То же	Курсор мыши выведен за пределы текущего элемента
onMouseOver	То же	Курсор мыши наведен на текущий элемент
onMouseUp	То же	Отпущена кнопка мыши в пределах текущего элемента
onMove	WINDOW	Перемещение окна
onReset	FORM	Сброс данных формы (щелчок по кнопке <input type="reset">)
onResize	WINDOW	Изменение размеров окна
onSelect	INPUT, TEXTAREA	Выделение текста в текущем элементе
onSubmit	FORM	Отправка данных формы (щелчок на кнопке <input type="submit">)
onUnload	BODY, FRAMESET	Попытка закрытия окна браузера и выгрузки документа

1.3. Динамический HTML

Консорциум W3C (<http://www.w3.org/dom>) определяет DHTML как спецификацию открытой объектной модели, которая обеспечивает полный доступ к документу и позволяет свободно манипулировать всем документом и его содержимым. Все элементы документа являются программируемыми объектами, управляемыми событиями мыши и клавиатуры. Благодаря DHTML такие операции, как добавле-

ние содержимого, изменение какой-либо части Web-страницы, не требуют обращения к серверу и перезагрузки страницы. DHTML расширяет возможности традиционного HTML, ориентированного в основном на оформление страниц, позволяет создавать страницы, которые могут в интерактивном режиме взаимодействовать с пользователем.

Одной из особенностей языка JavaScript является то, что на стороне клиента язык интегрирован с функциями браузера. Достигается это благодаря тому, что объектная модель браузера строится по принципу совместимости с объектной моделью JavaScript.

Все объекты браузера организованы в иерархическую структуру (рис. 1.1). Поскольку основные функции браузера реализуются в окне приложения, в котором отображается сам HTML-документ, центральным объектом иерархии является окно браузера. Оно представляется объектом `window`. Все другие объекты HTML рассматриваются как свойства этого объекта. На основе `window` можно определить объекты, свойства и методы, необходимые для полноценной работы с документами. Объекту `window` подчинены объекты следующего уровня, которые можно разделить на две группы:

- объекты браузера, предоставляющие доступ к свойствам, методам и событиям, происходящим в окне браузера:
 - `events`;
 - `history`;
 - `location`;
 - `navigator`;
 - `screen`;
- объекты документа и фреймов, позволяющие управлять элементами документов и фреймов, загруженных в браузер:
 - `document`;
 - `frames`.

Вторая группа объектов вместе с содержащимися в ней свойствами, методами и событиями образует объектную модель документа DOM (Document Object Model). Это все, что пользователь видит в окне документа: текст, ссылки, рисунки и т. д.

Для эффективного управления содержимым блоков HTML-страниц и их оформлением необходимо хорошо представлять себе иерархию объектов объектной модели. Вверху иерархии расположен самый старший класс `window`. Атрибуты и свойства этого класса относятся, как правило, ко всему окну в целом. Доступ к подчиненным классам и далее к семействам, элементам и атрибутам элементов осуществляется через dot-нотацию, т. е. через точку, как во многих объектно-ориентированных языках, например:

```
window.document.forms  
// все элементы семейства форм документа
```

Внутри объекту присваивается порядковый номер и может быть присвоено имя (идентификатор). Нумерация начинается с нуля (листинг 1.1).

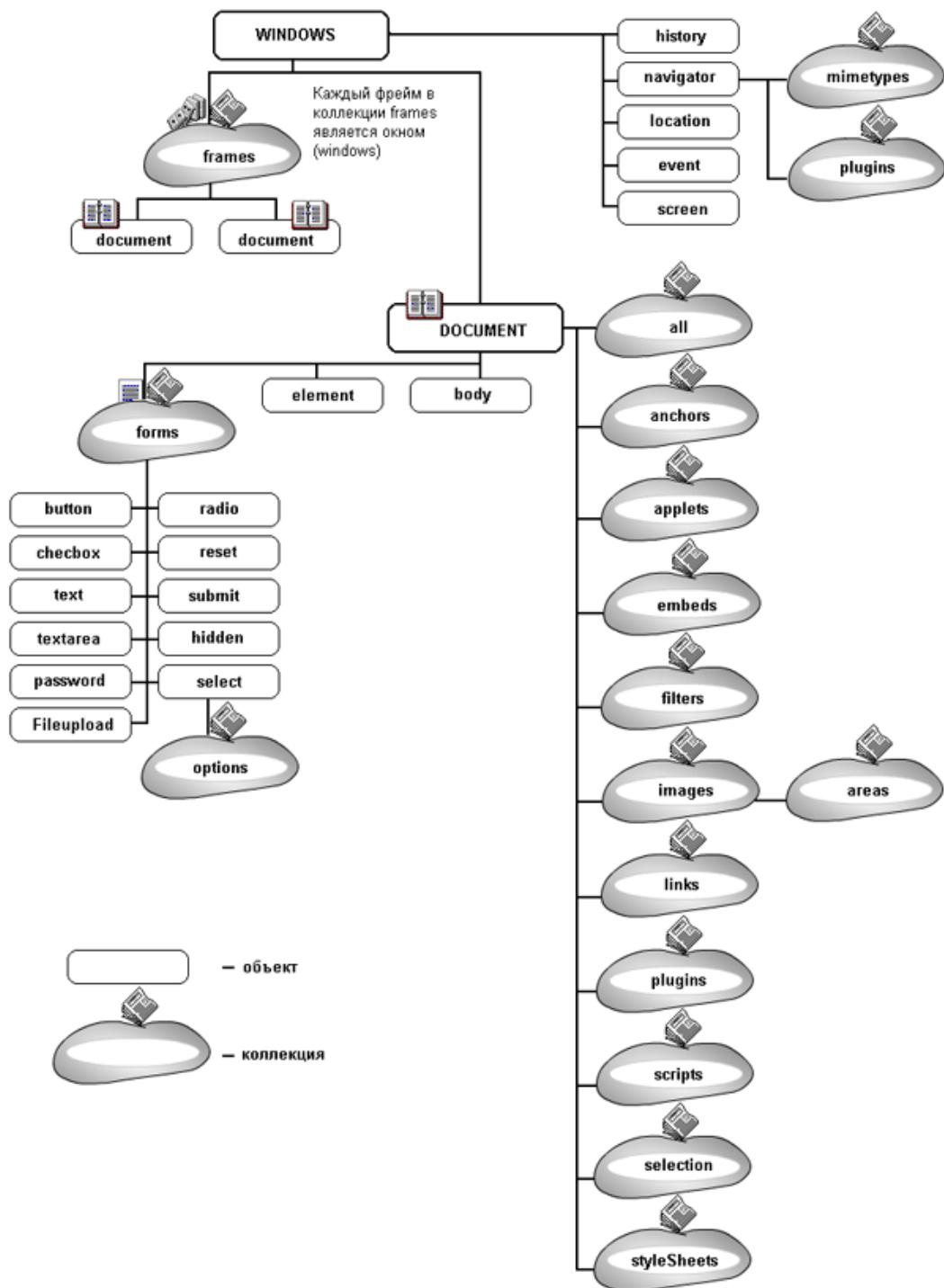


Рис. 1.1. Объектная модель

Листинг 1.1

```
window.document.forms[0]
// первая форма семейства форм
window.document.forms.form1
// форма семейства форм form1
window.document.forms.form1.elements.elements[1]
// второй элемент формы form1
window.document.forms.form1.elements.input1
// элемент input формы form1
```

Для изменения атрибутов HTML-элементов, необходимо указать путь к этому элементу и установить значение необходимого атрибута:

```
window.document.forms.form1.elements.input1.value="Значение 1";
window.document.forms.form1.elements.input1.style.color="red";
window.document.forms.form1.elements.input1.size=10;
```

Если элементу присвоен уникальный идентификатор (ID), доступ к элементу можно получить через ID:

```
window.document.getElementById("div1").style.display="none";
window.document.getElementById("div1").style.color="green";
```

В записи через dot-нотацию допускается опускать window, при этом запись будет относиться к текущему окну.

1.4. PHP — серверный язык программирования

PHP (Hypertext Preprocessor — препроцессор гипертекста) — это широко используемый язык сценариев общего назначения с открытым исходным кодом. PHP специально разработан для написания Web-приложений, исполняющихся на Web-сервере. Синтаксис языка во многом основывается на синтаксисе C, Java и Perl. Он очень похож на C и на Perl, поэтому для профессионального программиста не составит труда его изучить. С другой стороны, язык PHP проще, чем C, и его может освоить Web-мастер, не знающий пока других языков программирования.

Огромным преимуществом PHP, в отличие, например, от JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. PHP не зависит от быстродействия компьютера пользователя или его браузера, он полностью работает на сервере. Пользователь даже может не знать, получает ли он обычный HTML-файл или результат выполнения скрипта.

Сценарии на языке PHP могут исполняться на сервере в виде отдельных файлов, а могут интегрироваться в HTML-код страницы.

PHP способен генерировать и преобразовывать не только HTML-документы, но и изображения разных форматов (JPEG, GIF, PNG), файлы PDF и FLASH. PHP может формировать данные в любом текстовом формате, включая XHTML и XML.

PHP — кроссплатформенная технология. Дистрибутив PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix, Microsoft Windows, Mac OS и др. PHP поддерживается на большинстве Web-серверов, таких, как Apache, Microsoft Internet Information Server (IIS), Microsoft Personal Web Server и др.

Для большинства серверов PHP поставляется в двух вариантах: в качестве модуля и в качестве препроцессора CGI.

PHP поддерживает работу с ODBC и многими базами данных: MySQL, MSQQL, Oracle, PostgreSQL, SQLite и др.

Язык программирования PHP, особенно в связке с популярнейшей базой данных MySQL — оптимальный вариант для создания интернет-сайтов различной сложности. Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков Web-программирования.

1.5. СУБД MySQL

В настоящее время ни одно серьезное Web-приложение не может обойтись без взаимодействия с базой данных, обеспечивающей разнообразные возможности при работе с данными: сортировку, поиск, преобразование, редактирование и многое другое. При этом все низкоуровневые операции с файловой системой скрыты для программиста за несложными SQL-запросами. Есть множество различных видов баз данных, но мы будем рассматривать MySQL. Почему именно MySQL? Потому что она является небольшим, очень быстрым, компактным и простым в использовании сервером баз данных, идеальным для приложений малого и среднего размера.

1.5.1. Типы данных

Типы данных, применяемые в таблицах MySQL, можно разделить на следующие группы:

- целые числа;
- дробные числа;
- строки;
- бинарные данные;
- календарные (дата и время).

1.5.1.1. Целые числа

Общий вид указания целого числового типа данных:

предфикс INT [UNSIGNED]

Необязательный флаг UNSIGNED задает, что будет создано поле для хранения беззнаковых чисел (больших или равных нулю).

Типы целых числовых данных приведены в табл. 1.2.

Таблица 1.2. Целые числовые типы

Тип	Диапазон значений
TINYINT	-128 ... 127
SMALLINT	-32 768 ... 32 767
MEDIUMINT	-8 388 608 ... 8 388 607
INT	-2 147 483 648 ... 2 147 483 647
BIGINT	-9 223 372 036 854 775 808 ... 9 223 372 036 854 775 807

1.5.1.2. Дробные числа

MySQL поддерживает несколько типов дробных чисел (табл. 1.3).

В общем виде они записываются так:

ИмяТипа [(length, decimals)] [UNSIGNED]

где:

- length — количество знакомест (ширина поля), в которых будет размещено дробное число при его передаче;
- decimals — число знаков после десятичной точки, которые будут учитываться.

Таблица 1.3. Дробные числовые типы

Тип	Определение
UNSIGNED	Задает беззнаковые числа
FLOAT	Число с плавающей точкой небольшой точности
DOUBLE	Число с плавающей точкой двойной точности
REAL	Синоним для DOUBLE
DECIMAL	Дробное число, хранящееся в виде строки
NUMERIC	Синоним для DECIMAL

1.5.1.3. Строки

Строки представляют собой массивы символов. Обычно при поиске по текстовым полям по запросу SELECT не учитывается регистр символов, т. е. строки "Вася" и "ВАСЯ" считаются одинаковыми. Кроме того, если база данных настроена на автоматическую перекодировку текста при его помещении и извлечении, эти поля будут храниться в указанной вами кодировке. Для начала ознакомимся с типом строки, которая может хранить не более length символов, где length принадлежит диапазону от 1 до 255:

VARCHAR (length) [BINARY]

При занесении некоторого значения в поле такого типа из него автоматически вырезаются концевые пробелы. Если указан флаг `BINARY`, то при запросе `SELECT` строка будет сравниваться с учетом регистра.

Типы строковых данных приведены в табл. 1.4.

Таблица 1.4. Строковые типы

Тип	Максимальное число символов
VARCHAR	255
TINYTEXT	255
TEXT	65 535
MEDIUMTEXT	16 777 215
LONGTEXT	4 294 967 295

1.5.1.4. Бинарные данные

Бинарные данные почти аналогичны данным в формате `TEXT`, но только при поиске в них учитывается регистр символов.

Типы бинарных строковых данных приведены в табл. 1.5.

Таблица 1.5. Бинарные типы

Тип	Максимальное число символов
TINYBLOB	255
BLOB	65 535
MEDIUMBLOB	16 777 215
LONGBLOB	4 294 967 295

Бинарные данные не перекодируются автоматически, если при работе с установленным соединением включена возможность перекодирования текста "на лету".

1.5.1.5. Дата и время

MySQL поддерживает несколько типов полей, специально приспособленных для хранения даты и времени в различных форматах (табл. 1.6).

Таблица 1.6. Типы дата и время

Тип	Определение
DATE	Дата в формате ГГГГ-ММ-ДД
TIME	Время в формате ЧЧ:ММ:СС
DATETIME	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIMESTAMP	Дата и время в формате <code>timestamp</code> . Однако при получении значения поля оно отображается не в формате <code>timestamp</code> , а в виде ГГГГММДДЧЧММСС, что сильно уменьшает преимущества его использования в PHP

1.5.2. Таблицы MySQL

СУБД MySQL поддерживает в настоящее время несколько видов таблиц. Их можно разделить на два различных типа:

Транзакционные:

- InnoDB;
- BDB.

Без поддержки транзакций:

- HEAP;
- ISAM;
- MERGE;
- MyISAM.

Преимущества транзакционных таблиц (Transaction Safe Tables, TST):

- Высокая надежность. Даже если произойдет сбой в работе MySQL или возникнут проблемы с оборудованием, свои данные вы сможете восстановить либо методом автоматического восстановления, либо при помощи резервной копии и журнала транзакций.
- Можно сочетать несколько операторов и принимать их одной командой COMMIT.
- Внесенные изменения можно отменить командой ROLLBACK (если не установлен режим автоматической фиксации).
- Если произойдет сбой во время обновления, все изменения будут восстановлены (в нетранзакционных таблицах внесенные изменения нельзя отменить).

Преимущества таблиц без безопасных транзакций (Non Transaction Safe Tables, NTST):

- Работать с ними намного быстрее, т. к. не выполняются дополнительные транзакции.
- Для них требуется меньше дискового пространства.
- Для обновлений занято меньше памяти.

По умолчанию в MySQL принят тип таблиц MyISAM.

1.5.3. Структурированный язык запросов SQL

Структурированный язык запросов SQL позволяет выполнять различные операции с базами данных: создавать таблицы, вставлять, обновлять и удалять данные, проводить выборку данных и т. д. Рассмотрим основные операторы SQL.

CREATE DATABASE — эта команда создает новую базу данных:

```
CREATE DATABASE [IF NOT EXIST] db_name
```

Здесь db_name — имя новой создаваемой базы.

DROP DATABASE db_name — удаляет базу данных со всеми таблицами, входящими в ее состав; db_name — имя удаляемой базы.

- ❑ USE db_name — указывает MySQL, с какой базой данных вы намерены работать; db_name — имя выбираемой базы.
- ❑ CREATE TABLE — создает новую таблицу в выбранной базе данных. Синтаксис команды:

```
CREATE TABLE table_name [ (create_definitions,...) ] [table_options]
```

Здесь

- table_name — имя создаваемой таблицы;
- create_definitions — объявление столбца, его типа и атрибутов;
- table_options — тип таблицы.

Пример:

```
CREATE TABLE users (  
    id INT NOT NULL AUTO_INCREMENT, login VARCHAR (32) NOT NULL ,  
    password VARCHAR (12) NOT NULL , PRIMARY_KEY(id) ) TYPE=MyISAM
```

- ❑ DROP TABLE — удаляет одну или несколько таблиц:

```
DROP TABLE table_name1 [,table_name2,...]
```

Здесь table_name1, table_name2 — имена удаляемых таблиц.

- ❑ INSERT INTO... VALUES — вставляет новые записи в существующую таблицу базы данных:

```
INSERT INTO table_name VALUES (values1,...)
```

Пример:

```
INSERT INTO users VALUES ("1","user1","password1")
```

Порядок добавления столбцов можно устанавливать самостоятельно:

```
INSERT INTO users (password,id,login) VALUES ("password2","2","user2").
```

- ❑ INSERT INTO... SET — вставляет новые записи в существующую таблицу базы данных:

```
INSERT INTO table_name SET col_name1=value1[,col_name2=value2...]
```

Пример:

```
INSERT INTO users SET login="user3"
```

Поля, не указанные в запросе, получат значения по умолчанию, поле AUTO_INCREMENT получит значение на единицу больше, чем последнее.

- ❑ DELETE FROM — удаляет записи из таблицы базы данных:

```
DELETE FROM table_name [WHERE definition]
```

Удаление из таблицы table_name данных, удовлетворяющих указанным в definition условиям, и возвращает число удаленных записей.

Пример:

```
DELETE FROM users WHERE id>3
```

- ❑ SELECT — извлекает строки данных из одной или нескольких таблиц. Синтаксис команды:

```
SELECT column1,... [FROM table WHERE definition]  
[ORDER BY col_name [ASC|DESC],...] [LIMIT [offset], rows]
```

Здесь `column` — имя выбираемого столбца (для всех `*`). `WHERE` — определяет условия отбора строк. `ORDER BY` — сортирует строки по столбцу `col_name` в прямом (`ASC`) или обратном (`DESC`) порядке. `LIMIT` — сообщает MySQL о выводе только `rows` запросов, начиная с позиции `offset`.

- `UPDATE` — обновляет столбцы таблицы `table` в соответствии с их новыми значениями в строках существующей таблицы. Синтаксис:

```
UPDATE table SET col_name1=expr1[, col_name2=expr2...]
[WHERE definition]
[LIMIT rows]
```

В выражении `SET` указывается, какие именно столбцы следует изменить и какие значения. В `WHERE` определяется, какие строки подлежат изменению. `LIMIT` позволяет ограничить число изменяемых строк.

1.5.4. Функции PHP для работы с MySQL

Рассмотрим основные функции API для работы с MySQL из PHP-скриптов.

1.5.4.1. `mysql_connect`

Открывает соединение с сервером MySQL и возвращает его указатель или `false` при неудаче. Синтаксис функции:

```
resource mysql_connect([ string $server [, string $username [,string $password]]])
```

Это урезанный вариант синтаксиса функции `mysql_connect()`. Здесь рассмотрены три основные строковые (`string`) переменные, которых обычно хватает для работы.

- `$server` — сокет (хост), к которому производится подключение. Значение переменной не имеет никакого отношения к домену вашего сайта. Название и порт `$server` зависят от настроек самого сервера. Обычно эта переменная имеет значение `localhost`, что можно изменить в настройках PHP;
- `$username` — имя пользователя владельца процесса сервера;
- `$password` — пароль владельца процесса сервера.

1.5.4.2. `mysql_close`

Закрывает соединение с сервером MySQL. Синтаксис функции:

```
bool mysql_close ([resource link_identifier])
```

Возвращает `true` в случае успешного завершения, `false` при возникновении ошибки. `mysql_close()` закрывает соединение с базой данных MySQL, на которое указывает переданный указатель. Если параметр `link_identifier` не указан, закрывается последнее открытое (текущее) соединение. Непостоянные соединения автоматически закрываются в конце скрипта и `mysql_close()` не требуется.

1.5.4.3. `mysql_select_db`

Выбирает базу данных MySQL. Синтаксис функции:

```
bool mysql_select_db (string database_name [, resource link_identifier])
```

Возвращает `true` в случае успешного завершения, `false` при возникновении ошибки. `mysql_select_db()` выбирает для работы указанную базу данных на сервере, на который ссылается переданный указатель. Если параметр указателя опущен, используется последнее открытое соединение. Если нет ни одного открытого соединения, функция попытается соединиться с сервером аналогично функции `mysql_connect()`, вызванной без параметров. Каждый последующий вызов функции `mysql_query()` будет работать с выбранной базой данных.

1.5.4.4. `mysql_query`

Посыпает запрос MySQL. Синтаксис функции:

```
resource mysql_query ( string query [, resource link_identifier])
```

`mysql_query()` посыпает запрос активной базе данных сервера, на который ссылается переданный указатель. Если параметр `link_identifier` опущен, используется последнее открытое соединение. Если открытые соединения отсутствуют, функция пытается соединиться с СУБД, аналогично функции `mysql_connect()` без параметров. Результат запроса буферизируется. Только для запросов `SELECT`, `SHOW`, `EXPLAIN` и `DESCRIBE`, `mysql_query()` возвращает указатель на результат запроса, или `false`, если запрос не был выполнен. В остальных случаях, `mysql_query()` возвращает `true` при успешном запросе и `false` в случае ошибки. Значение, не равное `false`, свидетельствует лишь о том, что запрос был выполнен успешно, но не говорит о количестве затронутых или возвращенных рядов. Вполне возможна ситуация, когда успешный запрос не затронет ни одного ряда.

1.5.4.5. `mysql_fetch_row`

Обрабатывает ряд результата запроса и возвращает неассоциативный массив. Синтаксис функции:

```
array mysql_fetch_row (resource result)
```

Возвращает массив, содержащий данные обработанного ряда, или `false`, если рядов больше нет. `mysql_fetch_row()` обрабатывает один ряд результата, на который ссылается переданный указатель. Ряд возвращается в массиве. Каждая колонка располагается в следующей ячейке массива. Массив начинается с нулевого индекса. Последующие вызовы функции `mysql_fetch_row()` вернут следующие ряды или `false`, если рядов не осталось.

1.5.4.6. `mysql_fetch_assoc`

Обрабатывает ряд результата запроса и возвращает ассоциативный массив. Синтаксис функции:

```
array mysql_fetch_assoc (resource result)
```

Возвращает ассоциативный массив с названиями индексов, соответствующими названиям колонок или `false`, если рядов больше нет. Функция `mysql_fetch_assoc()` аналогична вызову функции `mysql_fetch_array()` со вторым параметром, равным `MYSQL_ASSOC`. Функция возвращает только ассоциативный массив. Если вам нужны как ассоциативные, так и численные индексы в массиве, обратитесь к функции

`mysql_fetch_array()`. Если несколько колонок в запросе имеют одинаковые имена, значение ключа массива с индексом названия колонок будет равно значению последней из колонок. Важно знать, что `mysql_fetch_assoc()` работает не медленнее, чем `mysql_fetch_row()`, предоставляя более удобный доступ к данным.

1.5.4.7. `mysql_fetch_array`

Обрабатывает ряд результата запроса, возвращая ассоциативный массив, численный массив или оба. Синтаксис функции:

```
array mysql_fetch_array (resource result [, int result_type])
```

Возвращает массив с обработанным рядом результата запроса, или `false`, если рядов больше нет. `mysql_fetch_array()` — это расширенная версия функции `mysql_fetch_row()`. В дополнение к хранению значений в массиве с численными индексами, функция возвращает значения в массиве с индексами по названию колонок. Важно знать, что `mysql_fetch_array()` работает не медленнее, чем `mysql_fetch_row()` и предоставляет более удобный доступ к данным.

1.5.4.8. `mysql_result`

Возвращает данные результата запроса. Синтаксис функции:

```
mixed mysql_result (resource result, int row [, mixed field])
```

`mysql_result()` возвращает значение одной ячейки результата запроса. Аргументом поля может быть смещение, имя поля, или имя поля и имя таблицы через точку (`tablename.fieldname`). Работая с большими результатами запросов, следует использовать одну из функций, обрабатывающих сразу целый ряд результата. Так как эти функции возвращают значение нескольких ячеек сразу, они намного быстрее `mysql_result()`. Вызовы функции `mysql_result()` не должны смешиваться с другими функциями, работающими с результатом запроса.

1.5.4.9. `mysql_num_rows`

Возвращает количество рядов результата запроса. Синтаксис функции:

```
int mysql_num_rows (resource result)
```

`mysql_num_rows()` работает только с запросами SELECT.

1.5.4.10. `mysql_insert_id`

Возвращает ID, сгенерированный при последнем запросе `INSERT`. Синтаксис функции:

```
int mysql_insert_id ([resource link_identifier])
```

`mysql_insert_id()` возвращает ID, сгенерированный колонкой с `AUTO_INCREMENT` последним запросом `INSERT` к серверу, на который ссылается переданный функции указатель `link_identifier`. Если параметр `link_identifier` не указан, используется последнее открытое соединение. `mysql_insert_id()` возвращает 0, если последний запрос не работал с полями `AUTO_INCREMENT`. Если вам нужно сохранить значение, убедитесь, что `mysql_insert_id()` вызывается сразу после запроса.

1.5.5. Работа с phpMyAdmin

Даже при виртуозном владении SQL и PHP работа по проектированию, построению и обновлению базы данных занимает много времени, если она выполняется при помощи штатных средств, входящих в дистрибутив MySQL. Значительно облегчить жизнь может Web-интерфейс для работы с MySQL — phpMyAdmin. Это приложение, написанное на PHP, может полностью управлять как целым сервером MySQL, так и отдельной базой данных или таблицей, быстро и легко осуществлять различные запросы. Для работы с MySQL не требуется знание SQL, интерфейс приложения переведен на множество языков, в том числе русский. phpMyAdmin может решать самые разнообразные задачи:

- создание и удаление баз данных;
- создание, удаление, переименование, копирование таблиц;
- удаление, изменение, добавление новых полей в таблицы;
- добавление, изменение, создание индексов;
- выполнение SQL-запросов;
- управление системными процессами сервера;
- управление учетными записями пользователей;
- экспорт/импорт данных;
- глобальный поиск по базе данных.

1.5.5.1. Запуск phpMyAdmin из Денвера

- phpMyAdmin входит в состав программной оболочки Денвер (см. разд. 1.6). Для запуска необходимо набрать в браузере <http://localhost/Tools/phpMyAdmin/> (Денвер должен быть включен), и вы увидите страницу, изображенную на рис. 1.2. Рассмотрим на примерах основные операции работы с phpMyAdmin.

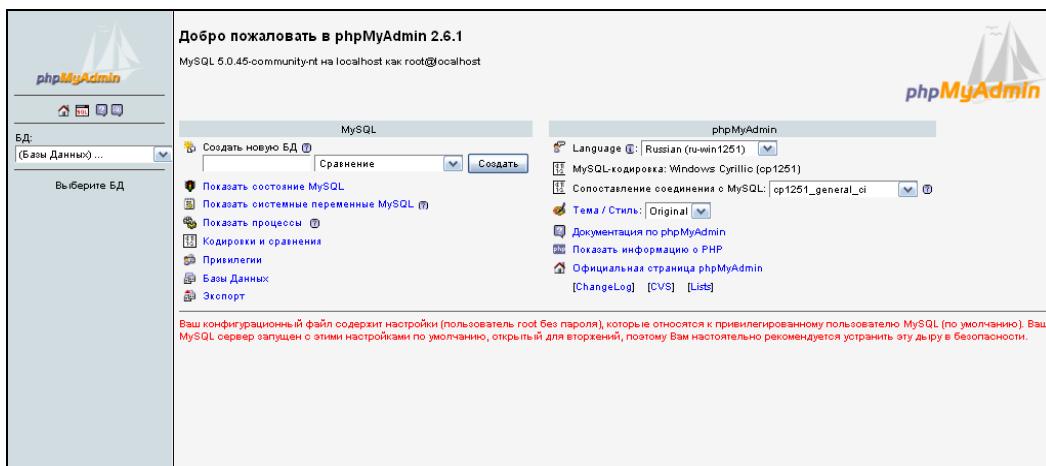


Рис. 1.2. Главная страница приложения phpMyAdmin

1.5.5.2. Создание базы данных

Для создания базы данных в форме, изображенной на рис. 1.2, введите название базы данных и выберите кодировку. Создадим базу данных my_test, кодировку выберем cp1251_general_ci. Далее нажмите кнопку **Создать**, при этом сразу начинается работа с базой данных под названием my_test (рис. 1.3).

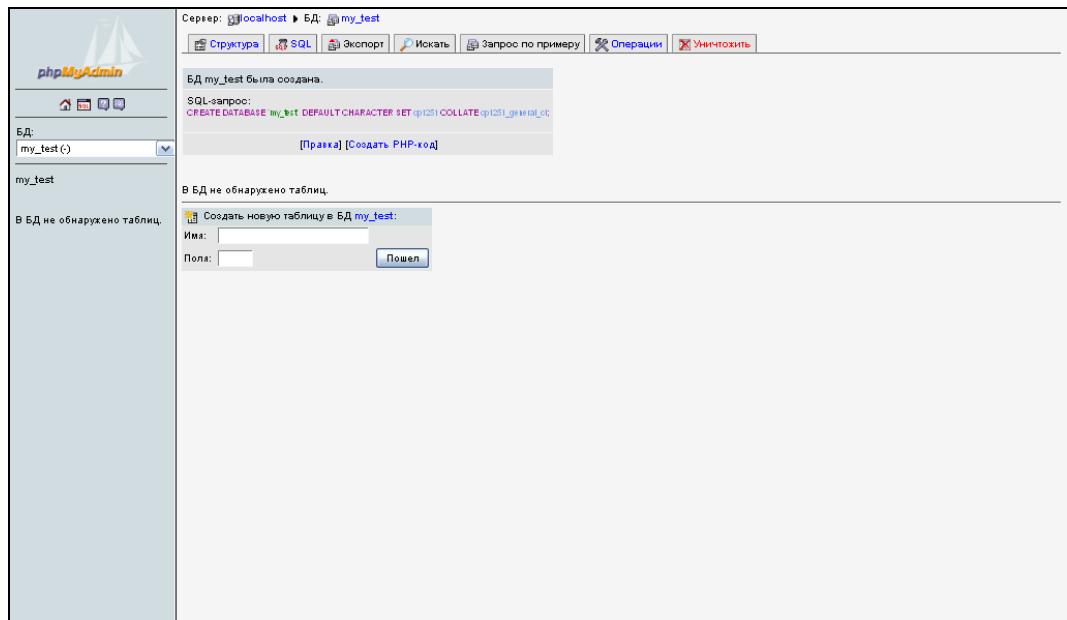


Рис. 1.3. База данных my_test

Итак, создана пустая база данных my_test. Теперь необходимо сформировать в ней таблицы.

1.5.5.3. Создание таблицы базы данных

Создадим в базе данных my_test таблицу my_table1. Заполним поля в форме создания базы на рис. 1.3: выберем имя — my_table1, зададим число полей — 4. Впоследствии мы сможем добавлять дополнительные поля или удалять ненужные. Нажимаем на кнопку **Пошел**. Переходим в форму создания структуры будущей базы (рис. 1.4). Для каждого поля необходимо ввести название, выбрать тип и значение по умолчанию. Первое поле id создадим уникальным и с дополнительным параметром AUTO_INCREMENT (рис. 1.5). Далее нажимаем кнопку **Сохранить**. Структура таблицы my_table1 базы данных my_test создана (рис. 1.6). Теперь можно приступать к ее заполнению.

Сервер: localhost > БД: my_test > таблица: my_table1

Поле	Тип	Длина/Значение*	Сравнение	Атрибуты	Ноль	По умолчанию**	Дополнительно
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		

Комментарий к таблице:

Тип таблицы: По умолчанию Сравнение:

Add 1 field(s) Помощь Сохранить

* Для типов поля "enum" и "set", введите значения по этому формату: 'a','b','c...'. Если вам понадобиться ввести обратную косую черту ("\\") или одинарную кавычку ("') среди этих значений, поставьте перед ними обратную косую черту (например, '\\yes' или 'a'b'). ** Для значений по умолчанию просто введите единственное значение без экранирования и квотирования, используя этот формат: a. Пожалуйста, введите значения для трансформации, используя этот формат: 'a','b','c...'. Если вам нужно поставить боксопечку(") или кавычку(""), экранируйте их (например 'Yes\b' or 'a\Vb'). Для списка доступных опций трансформации и трансформаций их MIME-типов кликните на описание трансформаций

Рис. 1.4. Форма создания структуры таблицы my_table1

Сервер: localhost > БД: my_test > таблица: my_table1

Поле	Тип	Длина/Значение*	Сравнение	Атрибуты	Ноль	По умолчанию**	Дополнительно
id	INT	9			not null		
data	DATETIME				not null		
name	VARCHAR	30			not null		
visible	SET	'yes','no'			not null	yes	

Комментарий Тип таблицы: Сравнение:

Add 1 field(s) Сохранить

* Для типов TINYTEXT и TEXT введите значения по этому формату: 'a','b','c...'. Если вам в поле BLOB понадобиться ввести обратную косую черту ("\\") или одинарную кавычку ("') среди этих значений, поставьте перед ними обратную косую черту (например, '\\yes' или 'a'b'). ** Для значений MEDIUMTEXT и LONGTEXT введите единственное значение без экранирования и квотирования, используя этот формат: a. Пожалуйста, введите значения для трансформации, используя этот формат: 'a','b','c...'. Если вам нужно поставить боксопечку(") или кавычку(""), экранируйте их (например 'Yes\b' or 'a\Vb'). Для списка доступных опций трансформации и трансформаций их MIME-типов кликните на описание трансформаций

Рис. 1.5. Заполнение структуры полей таблицы my_table1

Сервер: localhost > Бд: my_test > таблица : my_table1

Структура Обзор SQL Искать Вставить Экспорт Операции Очистить Уничтожить

таблица my_table1 была создана.

SQL запрос:

```
CREATE TABLE `my_table1`  
(`id` INT(9) NOT NULL AUTO_INCREMENT,  
 `data` DATETIME NOT NULL,  
 `name` VARCHAR(30) NOT NULL,  
 `visible` SET('yes', 'no') DEFAULT 'yes' NOT NULL,  
 UNIQUE  
 `id`);
```

[Правка] [Создать PHP-код]

	Поле	Тип	Сравнение	Атрибуты	Ноль	По умолчанию	Дополнительно	Действие
	id	int(9)			Да	NULL	auto_increment	
	data	datetime			Да			
	name	varchar(30)	cp1251_general_ci		Да			
	visible	set('yes', 'no')	cp1251_general_ci		Да	yes		

← Отметить все / Снять отметку со всех Сортировка:

Версия для печати Слоянный вид Предлагаемая структура таблицы

Add | field(s) В конец таблицы В начало таблицы После id Помощь

Индексы: Тип Количество элементов Действия Поле
id UNIQUE 0 id
Создать индекс на 1 колонках Помощь

Используемое пространство: Тип Использование
Данные 0 Bytes
Индекс 1,024 Bytes
Всего 1,024 Bytes

Статистика ряда:
Выражения Значение
Формат динамический
Сравнение cp1251_general_ci
Ряды 0
Далее Autoindex 1
Создание Июл 30 2010 г., 11:23
Последнее обновление Июл 30 2010 г., 11:23

Рис. 1.6. Структура таблицы my_table1 создана

1.5.5.4. Заполнение таблиц базы данных

Для вставки записей в таблицу базы данных, нажмите на ссылку **Вставить**. Появится форма вставки записи в таблицу базы данных (рис. 1.7).

Сервер: localhost > Бд: my_test > таблица: my_table1

[Структура](#) [Обзор](#) [SQL](#) [Искать](#) [Вставить](#) [Экспорт](#) [Операции](#) [Очистить](#) [Уничтожить](#)

Поле	Тип	Функция	Ноль	Значение
id	int(9)			
data	datetime			2010-07-30 11:28:47
name	varchar(30)			Имя 1
visible	set		..	yes no

Игнорировать

Поле	Тип	Функция	Ноль	Значение
id	int(9)			
data	datetime			2010-07-30 11:29:15
name	varchar(30)			Имя 2
visible	set		..	yes no

Вставить новый ряд И... Возврат
 Вставить новую запись

 [Помощь](#) [Переустановить](#)

Рис. 1.7. Форма вставки записей в таблицу my_table1

Заполним поля. Поле `id` можно не заполнять, т. к. оно уникальное (см. создание структуры) и значение `AUTO_INCREMENT` устанавливается автоматически. При нажатии

на кнопку **Пошел**, данные сохраняются в таблице. В появившемся окне нажимаем на ссылку **Обзор** и видим их в таблице (рис. 1.8).

The screenshot shows the phpMyAdmin interface for the 'my_test' database. The left sidebar lists databases (my_test, my_table1) and tables (my_table1). The main area displays the contents of 'my_table1' with two rows of data:

	id	data	name	visible
<input type="checkbox"/>	1	2010-07-30 11:28:47	Имя 1	yes
<input type="checkbox"/>	2	2010-07-30 11:29:15	Имя 2	yes

Below the table, there are buttons for 'Показать' (Show), 'Сортировать по ключу' (Sort by key), and 'Пошел' (Go). The bottom section contains links for 'Вставить новый ряд' (Insert new row), 'Версия для печати' (Print version), 'Распечатать (со всем текстом)' (Print (with all text)), and 'Экспорт' (Export).

Рис. 1.8. Просмотр записей в таблице my_table1

1.5.5.5. Экспорт-импорт баз данных

Часто возникает необходимость переноса баз данных с одного компьютера на другой. Операция импорта баз рассмотрена в разд. 1.6.4 при описании установки сайта на локальный компьютер пользователя. Рассмотрим операции экспорта.

This screenshot is similar to Figure 1.8, but the left sidebar shows a expanded list of databases. The 'magazin' database is selected, indicated by a blue background. Other databases listed include 'allrusia', 'altrunin', 'bn', 'board', 'book_najax1', 'catalog', 'country', 'doska', 'doska1', 'ekaterinburg', 'fonhome', 'hosting', 'hotel', 'information_schema', 'joomla', 'katalog_site', 'kmnneec', 'za071107', and 'my_test'. The main area shows the same table structure and data as Figure 1.8.

Рис. 1.9. Выбор базы для экспорта

Выберите базу для экспорта (рис. 1.9). Далее в появившемся окне (рис. 1.10) нажмите на ссылку **Экспорт**. Откроется страница экспорта (рис. 1.11). Заполняем данные. Устанавливаем флажок **послать**, вводим имя файла для экспорта. Помечаем таблицы для экспорта. Выбираем тип экспорта (INSERT, UPDATE или REPLACE) и нажимаем кнопку **Пошел**. Дамп базы данных сохранится в файле.

The screenshot shows the phpMyAdmin interface for the 'magazin' database. The left sidebar lists the database structure with tables like 'category', 'link_downloads', 'link_partners', etc. The main area displays a table of 13 tables with columns: таблица (Table), Действие (Action), Записи (Records), Тип (Type), Сравнение (Collation), Размер (Size), and Накладные расходы (Character Collate). Below the table, there are buttons for 'Отметить все' (Select All) and 'Снять отметку со всех' (Deselect All). At the bottom, there are buttons for 'Версия для печати' (Print Version), 'Создать новые таблицы в БД magazin:' (Create New Tables in DB magazin:), 'Имя:' (Name:), 'Поле:' (Field:), and 'Пошел' (Go).

Рис. 1.10. Список таблиц выбранной базы данных

The screenshot shows the 'Экспорт' (Export) dialog in phpMyAdmin. It lists tables: 'oplata_reg', 'rate', 'tovars', 'users', 'zakaz', and 'zakaz_table'. On the right, the 'Опции SQL' (SQL Options) panel is open with various checkboxes for exporting comments, structure, data, and specific options like AUTO_INCREMENT and backticks. A modal window titled 'Открытие «export_baza_magazin.sql»' (Opening «export_baza_magazin.sql») is displayed, asking if you want to open or save the file. The 'Сохранить файл' (Save File) option is selected. Other options include 'Выполнить автоматически для всех файлов данного типа' (Execute automatically for all files of this type) and 'OK' (OK) button. At the bottom, there are checkboxes for 'послать' (Send), 'Шаблон имени файла:' (File name template: 'export_baza_magazin'), 'Сжатие' (Compression), and 'Используйте __DB__ для имени БД, __TABLE__ для имени таблицы и любые strftime опции для задания времени, расширение будет добавлено автоматически. Любой другой текст будет сохранен.' (Use __DB__ for database name, __TABLE__ for table name and any strftime options for time, extension will be added automatically. Any other text will be saved.).

Рис. 1.11. Экспорт базы в файл export_baza_magazin.sql

1.6. Программная оболочка Денвер

Всю работу с сайтом будем проводить на локальном сервере, который вы можете установить и сконфигурировать у себя на компьютере. В этом случае, не придется покупать хостинг и иметь доступ в Интернет. Возможности локального сервера и хостинга практически ничем не отличаются, да и состоять сервер будет из тех же самых компонентов: база данных MySQL, сервер Apache, поддержка PHP и т. д. В принципе, все эти компоненты можно скачать по отдельности и, поставив их у себя на компьютере, правильно сконфигурировать, но ведь не все способны это сделать. И даже те, кто понимает, что к чему, могут испытать массу трудностей при установке и особенно при конфигурировании. Поэтому были придуманы и созданы разнообразные установочные пакеты, позволяющие при минимальных затраченных усилиях получить полностью рабочий и сконфигурированный локальный сервер. Вам останется только установить на него движок вашего сайта и начинать с ним работать, точно так, будто ваш сайт расположен на хостинге.

Варианты различных сборок локального сервера:

- Денвер;
- Xampp;
- VertrigoServ;
- Wamp.

Рассмотрим отладку сайта с помощью сервера Денвер. Это лично мой выбор, а вы можете пользоваться тем пакетом, который вам больше понравится. Работа с ними однотипна, и, поняв как пользоваться одним из пакетов, вы без труда разберетесь и с любой другой сборкой локального сервера.

1.6.1. Что такое Денвер?

Джентльменский набор Web-разработчика ("Д.н.в.р", читается "Денвер") — проект Дмитрия Котерова, набор дистрибутивов (Apache, PHP, MySQL, Perl и т. д.) и программная оболочка, используемые Web-программистами для разработки сайтов на "домашней" (локальной) Windows-машине без необходимости выхода в Интернет. Главная особенность Денвера — удобство при удаленной работе сразу над несколькими независимыми проектами и возможность размещения на Flash-накопителе. Он имеет нечто вроде ядра (или "сердца") — так называемый "базовый пакет", занимающий около 5,5 Мбайт. Все остальное поставляется в виде пакетов расширений. Официальный сайт: www.denwer.ru.

Базовый пакет содержит большинство необходимых программ и утилит:

- Инсталлятор (поддерживается также инсталляция на Flash-накопитель).
- Apache, SSL, SSI, mod_rewrite, mod_php.
- PHP5 с поддержкой GD, MySQL, sqLite.
- MySQL5 с поддержкой транзакций.
- Систему управления виртуальными хостами, основанную на шаблонах.

Чтобы создать новый хост, вам нужно лишь добавить папку в каталог /home, править конфигурационные файлы не требуется. По умолчанию уже поддержи-

ваются схемы именования каталогов многих популярных хостеров; новые можно без труда добавить.

- Систему управления запуском и завершением всех компонентов Денвера.
- phpMyAdmin — систему управления MySQL через Web-интерфейс.
- Эмулятор sendmail и SMTP-сервер (отладочная "заглушка" на localhost:25, складывающая приходящие письма в папку /tmp в формате EML); поддерживается работа совместно с PHP, Perl, Parser и т. д.

1.6.2. Получение дистрибутива и расширений Денвера

Для получения дистрибутива Денвера выполняем следующие процедуры:

1. Заходим на сайт и нажимаем на ссылку **Скачать Denwer 3** (рис. 1.12).

Перейдя по ссылке **Скачать Denver 3**, попадаем на страницу, где нам будет предложено зарегистрироваться, чтобы получить ссылку для скачивания Денвера (рис. 1.13).

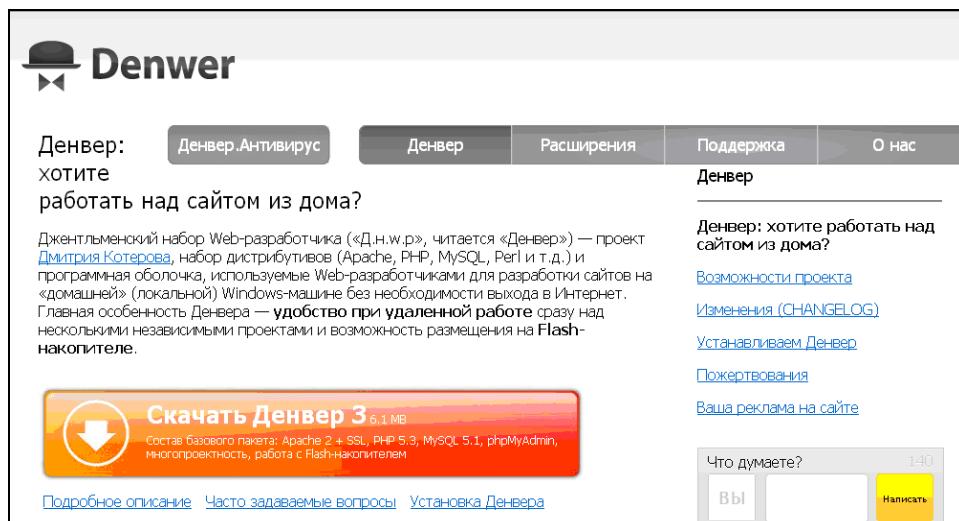


Рис. 1.12. Главная страница сайта www.denver.ru

2. Заполняем все поля и нажимаем на кнопку **Получить ссылку на скачивание**.

После этого появится окно с сообщением, что на ваш адрес электронной почты отправлена ссылка на скачивание Денвера (рис. 1.14).

Кроме базового пакета, имеется большое число расширений. В состав базового пакета Денвера, помимо стандартного набора модулей, входят только шесть библиотек: SQLite, iconv, GD2, MySQL и MySQLi, PDO. Другие дополнительные модули (например, PostgreSQL, mbstring, библиотеки PEAR и т. д.) поставляются в данном пакете расширения.

Зарегистрируйтесь

Введите Ваш E-mail, и мы вышлем на него ссылку для скачивания Денвера.

Ваше имя и фамилия:	<input type="text" value="New"/>	<input type="text" value="New"/>
Ваш E-mail:	<input type="text" value="kmvnews@bk.ru"/>	
<input checked="" type="checkbox"/> Присыпать мне новости проекта (не чаще 1 раза в месяц)		
Ваш краткий совет другим пользователям Денвера <i>New New:</i> _____		
Анатолий Малышев: Спасибо разработчикам Егор Петушкиов: Супер! John Konung: Спасибо! Елена Гриненко всем спасибо! Иван Хорхин: да я ток начинаю по этому жилаю всем удачам! Павел Демидов: спасибо 		
Получить ссылку на скачивание		

Рис. 1.13. Регистрация для получения ссылки на скачивание

Проверьте почту

На Ваш E-mail "kmvnews@bk.ru" выслана ссылка для скачивания Денвера.

Письмо обычно приходит в течение нескольких минут.
Если письма так и нет, проверьте на всякий случай папку Спам.

Рис. 1.14. Окно об отправке на e-mail ссылки на скачивание

Denwer

PHP 5: Денвер.Антивирус дополнительные модули	Денвер	Расширения	Поддержка	О нас
Модули PHP5				
PostgreSQL 8.4 Денвер + Parser				
PHP 4				
ActivePerl 5.8				
Parser 3				
ActiveDHTML 2.5				
Денвер				
PHP 5: дополнительные модули				
PostgreSQL 8.4 + модули PHP Денвер + Parser в одном пакете				
PHP 4: полный дистрибутив				
ActivePerl 5.8				
Parser 3 с поддержкой XML, SQL и электронной СМС				

[Скачать пакет расширения: модули PHP!](#)
Включает: REAR; модули для работы с различными СУБД;
Официальный сайт: <http://www.php.net>; Документация: <http://www.php.net/manual/ru/>

Рис. 1.15. Расширения Денвера

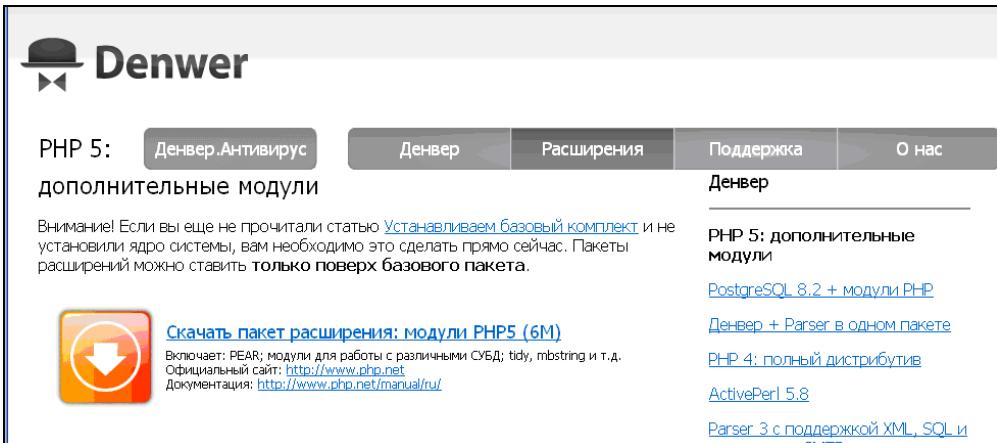


Рис. 1.16. Окно пакета расширений Денвера: модули PHP5

Скачаем из расширений модули PHP5. Для этого в пункте меню **Расширения** (см. рис. 1.12) нажмем на ссылку **Модули PHP5** (рис. 1.15).

В следующем окне нажимаем на ссылку **Скачать пакет расширения: Модули PHP5** (рис. 1.16).

ПРИМЕЧАНИЕ

Для получения ссылки на скачивание расширений Денвера также необходимо зарегистрироваться (см. рис. 1.13). Ссылка будет отправлена на указанный e-mail (см. рис. 1.14).

Далее переходим по ссылке, пришедшей в письме на ваш e-mail, и попадаем на страницу загрузки базового пакета Денвера (рис. 1.17). Нажимаем на ссылку **нажмите сюда** и в открывшемся окне (рис. 1.18) — на кнопку **Сохранить файл**. Начнется скачивание файла на ваш компьютер.

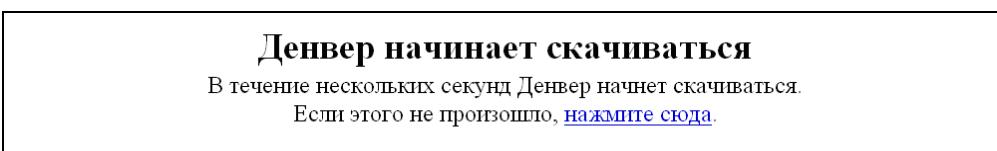


Рис. 1.17. Окно скачивания программного файла Денвера

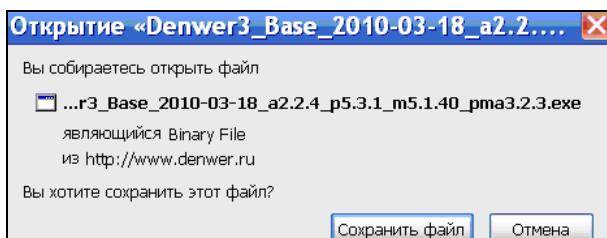


Рис. 1.18. Скачивание программного файла Денвера

1.6.3. Установка Денвера

Запускаем скачанный файл. Запускается программа установки комплекса Денвер (рис. 1.19).

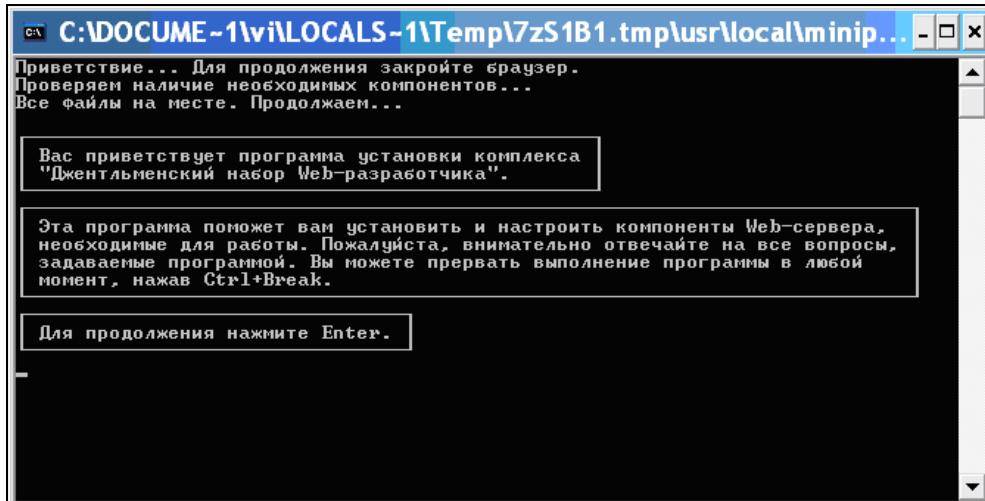


Рис. 1.19. Запуск программы установки Денвера

Программа предлагает выбрать каталог установки Денвера. По умолчанию установка будет произведена в c:\WebServers (рис. 1.20). Выбираем папку установки и нажимаем клавишу <Enter>.

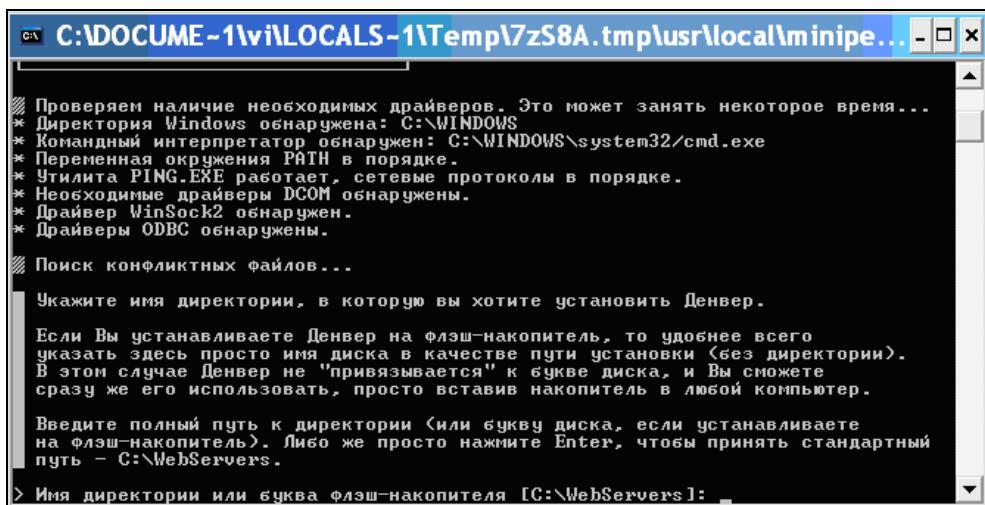


Рис. 1.20. Выбор каталога для установки Денвера

Инсталлятор создаст отдельный виртуальный диск, который необходим для функционирования всех компонентов системы (рис. 1.21). Такой диск сильно упрощает работу с Web-инструментарием, позволяя устроить на машине нечто вроде "маленького Unix".

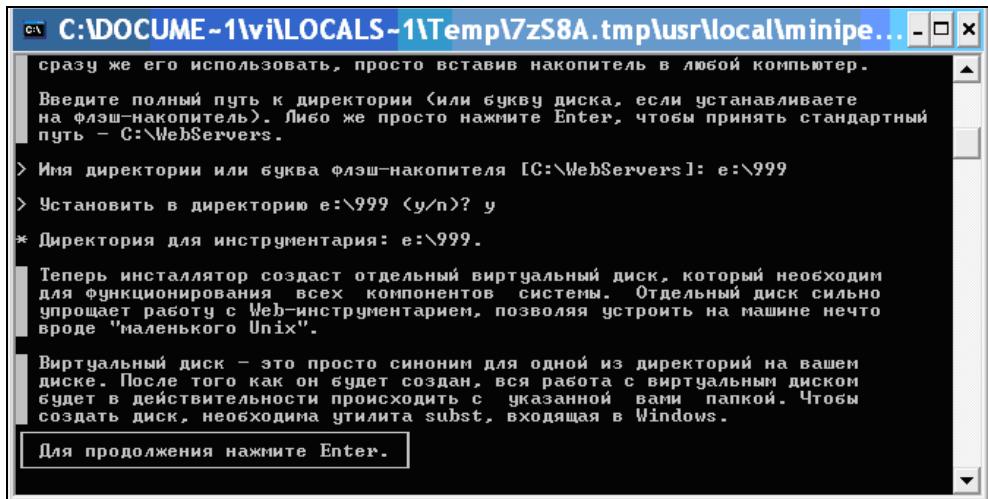


Рис. 1.21. Информация о создании виртуального диска

По умолчанию программа предлагает диск Z (рис. 1.22), скорее всего, диск с таким именем свободен. Впрочем, можно ввести и любую другую букву диска, который еще не занят. Существующие диски указывать нельзя.

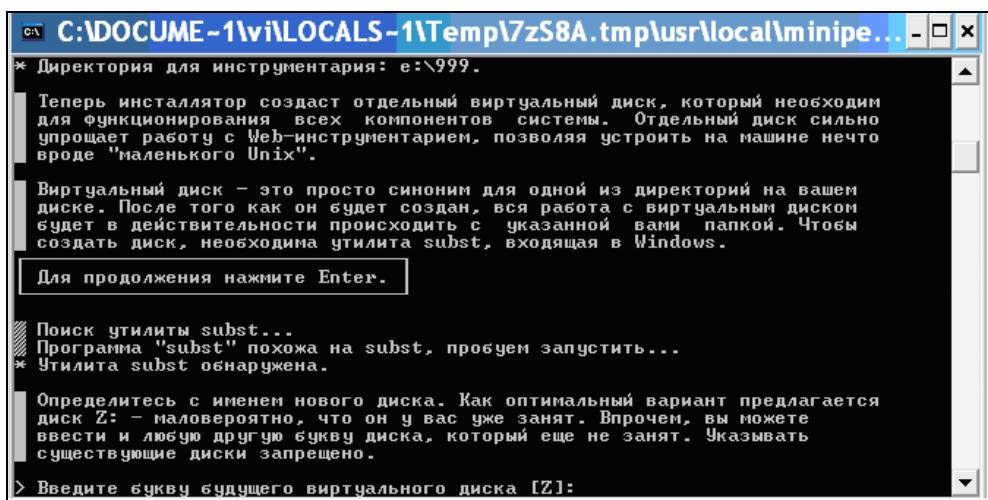


Рис. 1.22. Выбор буквы при создании виртуального диска

После выбора диска программа копирует файлы в выбранную папку установки. Денвер может запускаться в двух режимах:

1. Виртуальный диск создается при загрузке ОС. Запуск серверов осуществляется с помощью ярлыка на Рабочем столе. При завершении работы Денвера виртуальный диск не отключается. Этот режим целесообразен, если вы собираетесь использовать виртуальный диск, не запуская серверов (например, хотите запускать Perl-скрипты не только из браузера, но и из командной строки).
2. При загрузке ОС виртуальный диск не создается. На Рабочем столе так же, как и в п. 1, создаются ярлыки для запуска и останова серверов. При запуске серверов вначале создается виртуальный диск, после останова диск отключается. Необходимо помнить, что в этом режиме при неактивном Денвере доступа к виртуальному диску (в частности, к Perl) не будет. Кроме того, некоторые версии Windows не умеют правильно отключать виртуальный диск (требуется перезагрузка).

Установщик рекомендует выбрать вариант 1, потому что он наиболее удобен (рис. 1.23).

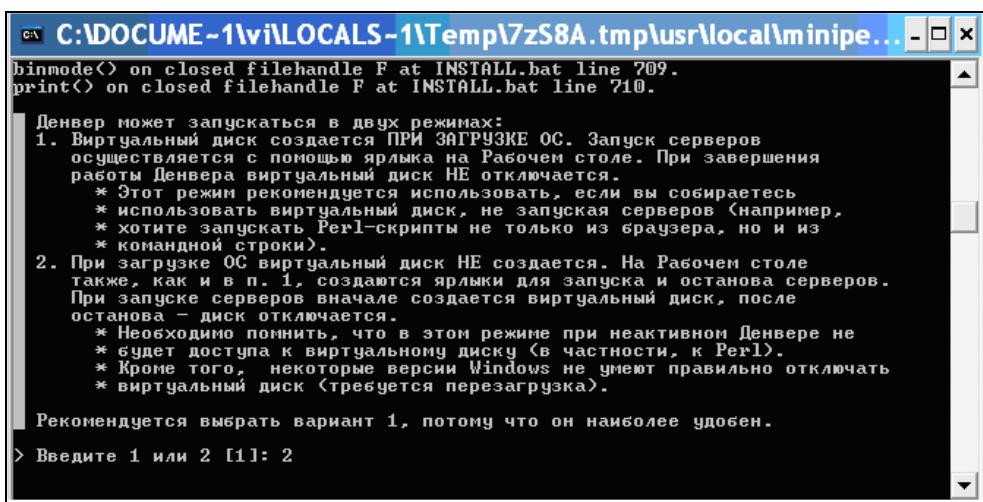


Рис. 1.23. Выбор режима запуска Денвера

После выбора варианта установки, программа предлагает создать ярлыки для запуска Денвера на рабочем столе (рис. 1.24). Запуск Денвера осуществляется программой **/denwer/Run.exe**, останов — **/denwer/Stop.exe**, перезапуск — **/denwer/Restart.exe**. Установка Денвера завершена (рис. 1.25).

Проверить работоспособность Денвера можно, набрав в адресной строке браузера **http://localhost**. В случае успешной установки вы увидите страницу, изображенную на рис. 1.26.

Я думаю, не составит труда установить и пакет дополнительных модулей PHP5 для Денвера. Программа просит указать каталог установки базового пакета Денвер.

После успешной установки вам необходимо открыть файл `/usr/local/php5/php.ini` в любом текстовом редакторе и раскомментировать директивы подключения тех или иных модулей, чтобы они выглядели так:

```
extension = имя_модуля
```

Модули, закомментированные при помощи двойной точки с запятой `; ;`, как правило, требуют дополнительных внешних библиотек и не работают в конфигурации по умолчанию. Будьте осторожны при их подключении!

The terminal window shows the following text:

```
C:\DOCUMENTS\LOCALS\Temp\7zS8A.tmp\usr\local\minipe...
binmode() on closed filehandle F at INSTALL.bat line 709.
print() on closed filehandle F at INSTALL.bat line 710.

Денвер может запускаться в двух режимах:
1. Виртуальный диск создается ПРИ ЗАГРУЗКЕ ОС. Запуск серверов
осуществляется с помощью ярлыка на Рабочем столе. При завершении
работы Денвера виртуальный диск НЕ отключается.
* Этот режим рекомендуется использовать, если вы собираетесь
* использовать виртуальный диск, не запуская серверов (например,
* хотите запускать Perl-скрипты не только из браузера, но и из
* командной строки).
2. При загрузке ОС виртуальный диск НЕ создается. На Рабочем столе
также, как и в п. 1, создаются ярлыки для запуска и останова серверов.
При запуске серверов вначале создается виртуальный диск, после
останова - диск отключается.
* Необходимо помнить, что в этом режиме при неактивном Денвере не
* будет доступа к виртуальному диску (в частности, к Perl).
* Кроме того, некоторые версии Windows не умеют правильно отключать
* виртуальный диск (требуется перезагрузка).

Рекомендуется выбрать вариант 1, потому что он наиболее удобен.

> Введите 1 или 2 [1]: 2
> Создать ярлыки на Рабочем столе для запуска Денвера (y/n)?
```

Рис. 1.24. Создание ярлыков для запуска Денвера

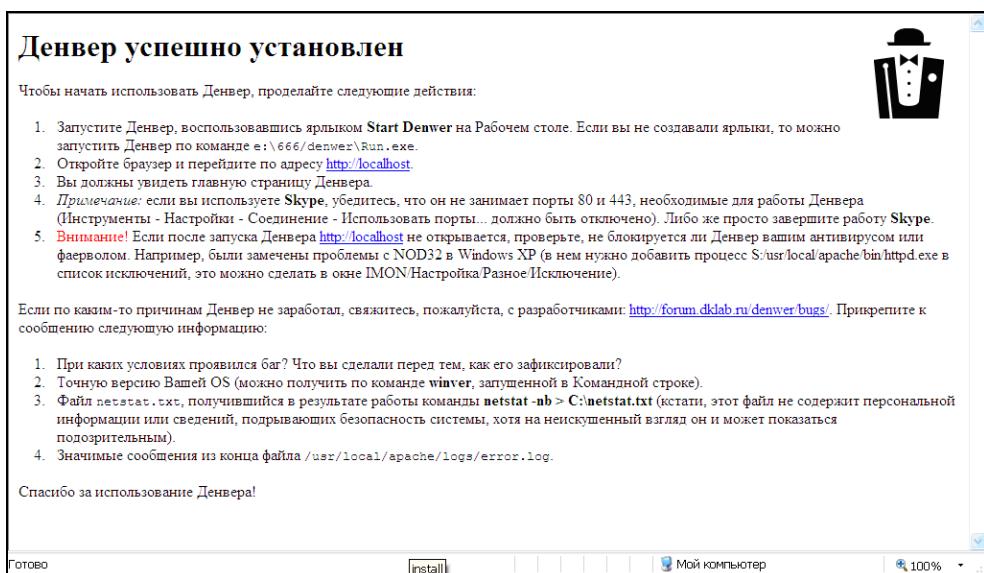


Рис. 1.25. Завершение установки Денвера

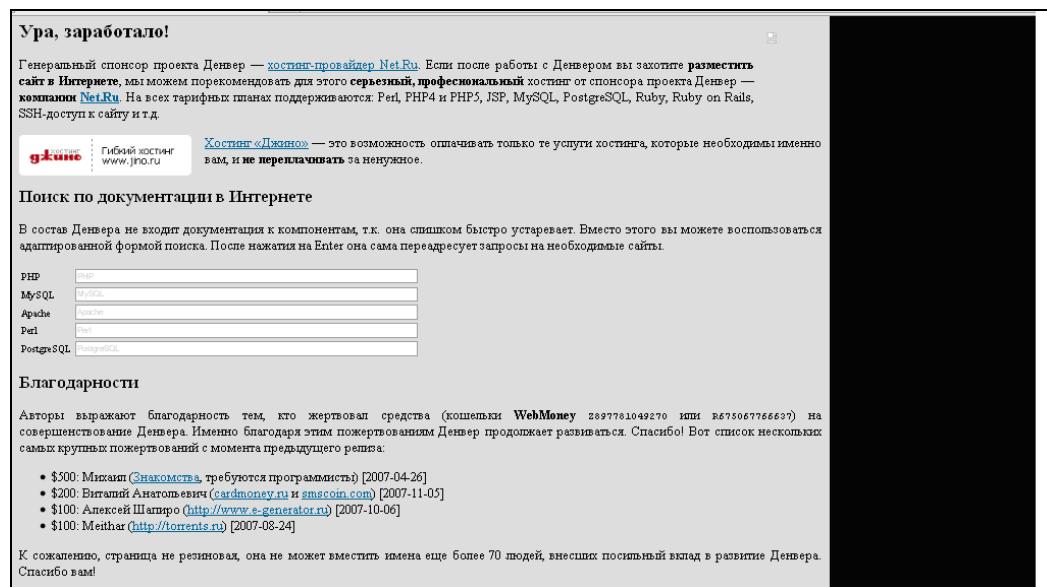


Рис. 1.26. Проверка работоспособности Денвера

Рекомендуется не подключать все модули сразу, а делать это позже, по мере необходимости. Такой подход позволит сэкономить оперативную память и сделает работу сервера более стабильной.

Пакет дополнительных модулей PHP5 содержит набор скриптов для работы с PEAR — каталогом разнообразных модулей для PHP. PEAR инициализируется при помощи bat-файла **/usr/local/php5/go-pear.bat**, который нужно запустить на исполнение. Конечно, содержать все библиотеки PEAR пакет не может, т. к. их очень много. Здесь ситуация похожа на работу с модулями Perl: в дистрибутиве поставляются лишь наиболее употребительные библиотеки, а также инсталлятор, позволяющий интерактивно установить остальное. Если вам нужен какой-нибудь "нестандартный" модуль, имеющийся на официальном сайте PEAR, воспользуйтесь для его установки утилитой **/usr/local/php5/pear.bat**. Конечно, утилиту следует запускать уже после того, как PEAR был инициализирован.

После установки и настройки пакета не забудьте перезапустить Денвер!

1.6.4. Размещаем сайт на локальном компьютере

Для создания сайта на локальном компьютере необходимо в каталоге установки Денвера в папке `/home` создать папку с названием нашего сайта, например `magazin`. Далее в папке `magazin` создаем папку `www`, которая будет корневой папкой нашего сайта. В нее копируем с диска все содержимое папки `magazin`. Затем нужно загрузить файлы баз данных. Это можно сделать двумя способами:

- создание базы данных и импорт дампов таблиц через phpMyAdmin;
- прямой записью файлов баз данных.

The screenshot shows the phpMyAdmin 2.6.1 interface. On the left, there's a sidebar with a logo and navigation links for 'Самые популярные' (Popular), 'Начальная страница' (Home), 'Последние загрузки' (Recent uploads), 'Katalogs' (Catalogs), 'английский' (English), and 'Russian'. Below this is a search bar and a link to 'Система электр...'. The main area has tabs for 'localhost' and 'Мульти-портал...'. The 'localhost' tab is active, showing the title 'Добро пожаловать в phpMyAdmin 2.6.1' and the MySQL version 'MySQL 5.0.45-community-nt на localhost как root@localhost'. A form on the right allows creating a new database with fields for 'БД:' (Database) set to '(Базы Данных)...' and 'Name:' set to 'magazin'. The 'charset' dropdown is set to 'cp1251_bin'. A 'Создать' (Create) button is visible. To the right of the form is a sidebar with various links: 'Показать состояние MySQL', 'Показать системные переменные MySQL', 'Показать процессы', 'Кодировки и схематика', 'Привилегии', 'Базы Данных', and 'Экспорт'. The 'Language' dropdown is set to 'Russian (ru-win1251)'. Other settings include 'MySQL-хардвера: Windows Cyrillic (cp1251)', 'Соответствие соединения с MySQL:', 'Тема / Стиль: Original', 'Документация по phpMyAdmin', 'Показать информацию о PHP', and 'Официальная страница phpMyAdmin [ChangeLog] [CVS] [List]'. A note at the bottom states: 'Ваш конфигурационный файл содержит настройки (пользователь root без пароля), которые относятся к привилегированному пользователю MySQL (по умолчанию). Ваш MySQL сервер запущен с этими настройками по умолчанию, открытый для вторжений, поэтому Вам настоятельно рекомендуется устраниить эту дыру в безопасности.' At the bottom left is a 'Готово' (Done) button.

Рис. 1.27. Создание новой базы данных в phpMyAdmin

This screenshot shows the phpMyAdmin interface after the database 'magazin' was created. The left sidebar shows the 'БД:' (Database) section with '(Базы Данных)...' selected. The main area shows the message 'БД magazin была создана.' (Database magazin was created.) Below it is the SQL query: 'CREATE DATABASE `magazin` DEFAULT CHARACTER SET cp1251 COLLATE cp1251_general_ci;'. There are buttons for '[Правка]' (Edit) and '[Создать PHP-код]' (Create PHP code). The message 'В БД не обнаружено таблиц' (No tables found in DB) is displayed. At the bottom, there's a form to 'Создать новую таблицу в БД magazin': 'Имя:' (Name:) with an input field, 'Поля:' (Fields:) with an input field, and a 'Помощь' (Help) button.

Рис. 1.28. База данных magazin создана

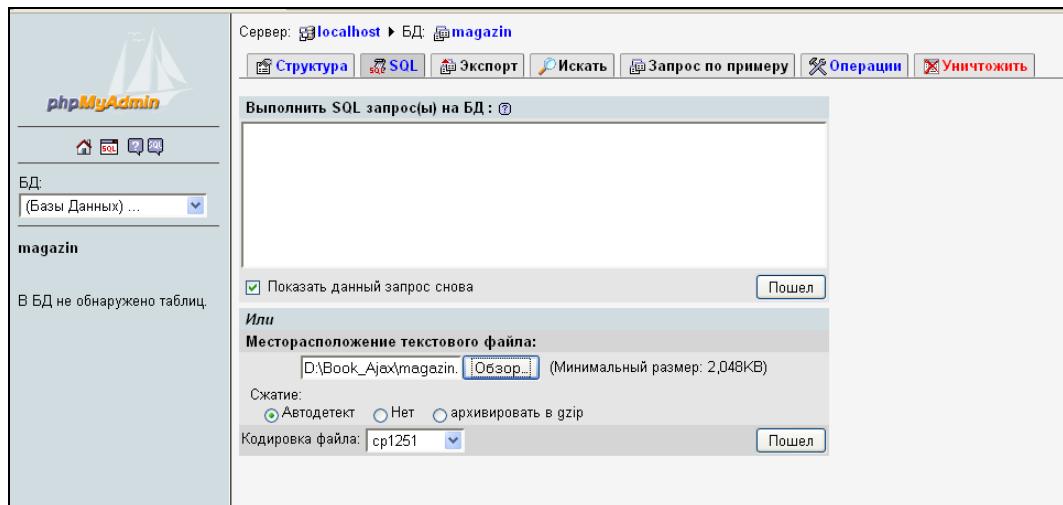


Рис. 1.29. Импорт дампа базы данных

- Для загрузки базы данных через phpMyAdmin, запускаем Денвер и набираем в адресной строке **http://localhost/Tools/phpMyAdmin** (рис. 1.27). Пишем название новой базы данных (*magazine*) и нажимаем кнопку **Создать**. База данных создана (рис. 1.28).

Далее импортируем дамп. Для этого нажимаем на ссылку **SQL** или **Импорт** в зависимости от версии MySQL и на открывшейся странице (рис. 1.29) указываем путь к дампу базы данных (дамп базы данных *magazin* расположен на прилагаемом компакт-диске *magazin/magazin.sql*), устанавливаем кодировку согласно рис. 1.29 и нажимаем на кнопку **Пошел**. При успешном импорте откроется страница, изображенная на рис. 1.30. Как видно из рисунка, загружено 13 таблиц.

При прямом копировании файлов, перепишите все содержимое папки *sql* на прилагаемом компакт-диске в папку *\usr\local\mysql5\data* каталога установки Денвера. Перезапустите Денвер. Войдите в phpMyAdmin, и увидите две базы данных — *magazin* и *book_examples*.

Для создания сайта с примерами на локальном компьютере необходимо в каталоге установки Денвера в папке */home* создать папку с названием нашего сайта *book_primers*. Далее в папке *book_primers* создаем папку *www*, которая будет корневой папкой нашего сайта примеров. В нее копируем с прилагаемого компакт-диска все содержимое папки *book_primers*.

Для загрузки сайта наберите в адресной строке **http://magazin**. Сайт успешно запущен (рис. 1.31). Для запуска примеров к книге наберите в адресной строке **http://book_primers** и на открывшейся странице (рис. 1.32) выберите нужный пример, например *2_7* (рис. 1.33).

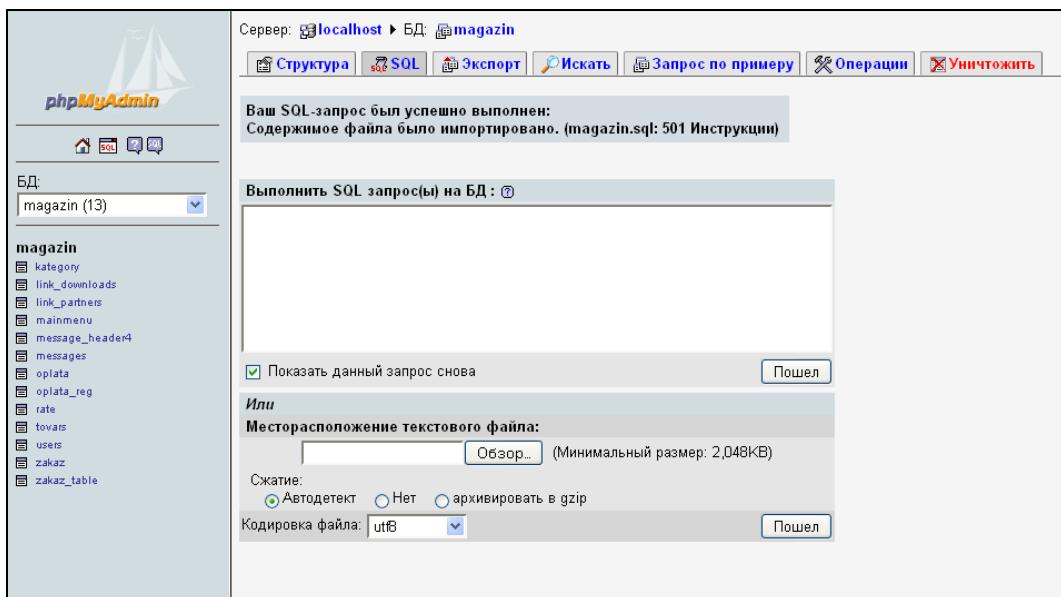


Рис. 1.30. Импорт дампа базы данных выполнен успешно

The screenshot shows a local website's upload interface. At the top, there are navigation links: Товары, Заказы, Переписка, Оплата, Регистрация, and user info (Логин: 11111, Пароль: *****, Вход ->, Нет сообщений). The main content area has several sections: 'Заголовок' (All products (7), Scripts (0), Database files (7), Region cards (0)), 'АКЦИЯ !!!' (Base of postal indices of populated points of Russia 100.00 - 95.00 rub.), 'Единый федеральный реестр туроператоров 50.00 - 45.00 руб.', 'Статистика' (On 01.06.2010: Tovar - 7, Polzovately - 82, Zakaz - 28, Zakazov oplichennykh - 11), 'Контакты' (Email: 385771293, my_shop@bk.ru), 'Поиск товаров' (Search bar, 'Название', 'Поиск в', 'в названии' checked, 'в описании'), 'Все товары' (List of products), 'Телефонные коды городов России' (List of phone codes), 'Курсы валют' (Currency rates: 30.862 rub, 44.858 usd, 6.48 usd, 4.46 eur), 'Корзина' (Cart icon, 1 item: 100.00 rub), 'У партнеров' (Business partners), and 'Контакты' (Contact information).

Рис. 1.31. Страница загрузки сайта на локальном компьютере

Index of /

Name	Last modified	Size	Description
1/	13-Jul-2010 11:05	-	
2-1/	29-Jul-2010 09:17	-	
2-2/	10-Jul-2010 18:20	-	
2-3/	10-Jul-2010 18:58	-	
2-4/	02-Sep-2010 14:27	-	
2-5/	20-Jul-2010 19:42	-	
2-6/	22-Jul-2010 15:33	-	
2-7/	25-Jul-2010 03:08	-	
2-8/	29-Jul-2010 12:25	-	
book_ajax.ppr	03-Sep-2010 12:13	1.9K	

Apache/2.2.4 (Win32) mod_ssl/2.2.4 OpenSSL/0.9.8d PHP/5.2.4 Server at book_examples Port 80

Рис. 1.32. Загрузка страницы с примерами

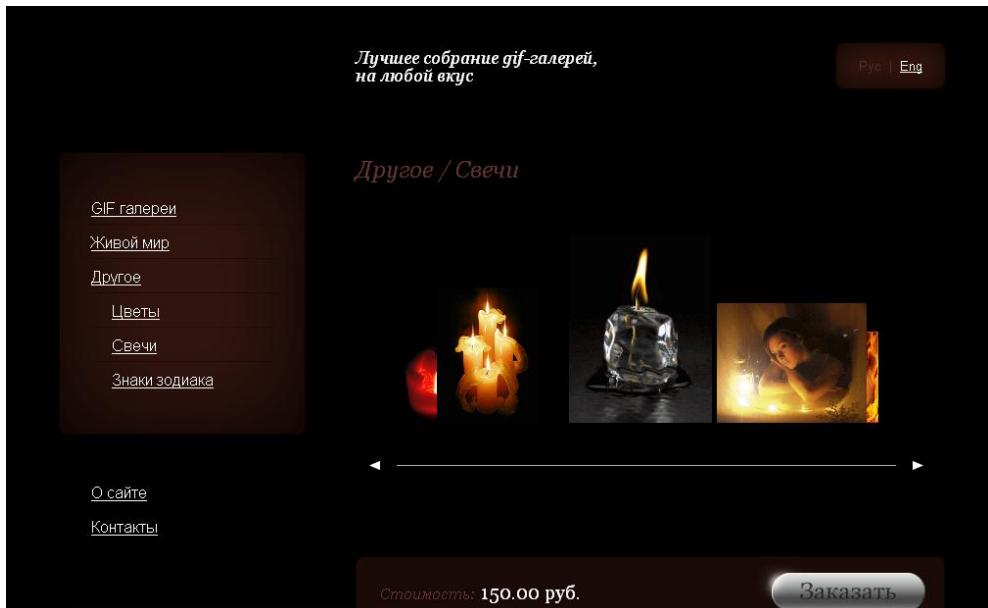


Рис. 1.33. Выбран пример 2_7



Глава 2

Технология AJAX

Когда имеющихся возможностей становится мало, а совершенствовать существующее уже некуда, тогда и происходит технологический прорыв. Таким прорывом и стал AJAX (Asynchronous JavaScript and XML) — подход к построению пользовательских интерфейсов Web-приложений, при котором Web-страница, не перезагружаясь, сама догружает нужные пользователю данные.

2.1. Что такое AJAX

Так что же такое AJAX? Впервые об AJAX заговорили после появления в феврале 2005 года статьи Джесси Джеймса Гарретта (Jesse James Garrett) "Новый подход к Web-приложениям". AJAX — это не самостоятельная технология, а идея, которая базируется на двух основных принципах:

- динамическое изменение содержания страницы с помощью DHTML;
- обращение к серверу "на лету" через объект XMLHttpRequest.
- Это позволяет при создании Web-приложений реализовать функциональность, которая раньше была доступна только прикладным программам: всевозможную анимацию, общение с сервером без перезагрузки страницы, использование "не Web"-элементов (деревьев, табов, сплиттеров и др.). Технология AJAX стала наиболее популярна после того, как компания Google начала активно использовать ее при разработке своих сайтов, таких как Gmail, Google maps и Google suggest. Создание этих сайтов подтвердило эффективность данного подхода. И задача разработчиков — взять на вооружение все, что позволит создавать новые приложения, которые соответствуют новым, пока еще правда "сырым", стандартам, объединяемым звучным названием "Web 2.0".

Типичное AJAX-приложение состоит как минимум из двух частей: первая выполняется в браузере и написана, как правило, на JavaScript, а вторая находится на сервере и написана, например, на PHP. Между этими двумя частями происходит обмен данными через механизм XMLHttpRequest.

2.1.1. Обмен данными между клиентом и сервером

Для того чтобы осуществлять обмен данными, на Web-странице должен быть создан объект XMLHttpRequest, который является своеобразным посредником между

браузером пользователя и сервером. С помощью XMLHttpRequest можно отправить запрос на сервер, а также получить ответ в виде различного рода данных. Объект XMLHttpRequest создается следующим образом:

- для Internet Explorer:

```
var xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
либо
var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP").
```

- для других браузеров:

```
var xmlhttp = new XMLHttpRequest();
```

Листинг 2.1 содержит код для всех браузеров.

Листинг 2.1

```
if (window.XMLHttpRequest)
{
    Request = new XMLHttpRequest();
}
else if (window.ActiveXObject)
{
    try
    {
        Request = new ActiveXObject("Microsoft.XMLHTTP");
    }
    catch (CatchException)
    {
        Request = new ActiveXObject("Msxml2.XMLHTTP");
    }
}
```

В конце этого процесса xmlhttp должен ссылаться на корректный объект XMLHttpRequest, независимо от используемого пользователем браузера.

2.1.2. Свойства и методы объекта XMLHttpRequest

Свойства объекта XMLHttpRequest:

- onreadystatechange — одно из самых главных свойств объекта XMLHttpRequest, с помощью которого задается обработчик, вызываемый всякий раз при смене статуса объекта;
- readyState — число, обозначающее статус объекта:
- 0 — объект не инициализирован;
 - 1 — объект загружает данные;
 - 2 — объект загрузил свои данные;
 - 3 — объект не полностью загружен, но может взаимодействовать с пользователем;
 - 4 — объект полностью инициализирован; получен ответ от сервера;
- responseText — представление ответа сервера в виде обычного текста (строки);

- `responseXML` — объект документа, совместимый с DOM, полученной от сервера;
- `status` — состояние ответа от сервера;
- `statusText` — текстовое представление состояния ответа от сервера.

Опираясь на состояние готовности объекта (`readyState`), можно предоставить посетителю информацию о том, на какой стадии находится процесс обмена данными с сервером и, возможно, оповестить его об этом визуально.

Методы объекта `XMLHttpRequest`:

- `abort()` — отменить текущий запрос к серверу;
- `getAllResponseHeaders()` — получить все заголовки ответа от сервера;
- `getResponseHeader("имя_заголовка")` — получить указанный заголовок;
- `open ("тип_запроса", "URL", "асинхронный", "имя_пользователя", "пароль")` — инициализировать запрос к серверу с указанием метода запроса. Тип запроса и URL — обязательные параметры. Третий аргумент — булево значение. Обычно всегда указывается `true` или не указывается вообще (по умолчанию — `true`). Четвертый и пятый аргументы служат для аутентификации (это очень небезопасно, хранить данные об аутентификации в скрипте, т. к. код может посмотреть любой пользователь);
- `send("содержимое")` — послать HTTP запрос на сервер и получить ответ;
- `setRequestHeader("имя_заголовка", "значение")` — установить значения заголовка запроса.

2.1.3. Запрос к серверу и обработка ответа

Для запроса к серверу нужно следовать одной и той же основной схеме практически во всех ваших AJAX-приложениях:

- получить какие-либо данные (например, из Web-формы);
- создать URL для подключения;
- открыть соединение с сервером;
- установить функцию для сервера, которая выполнится после его ответа;
- передать запрос.

Например, при событии `onchange` элемента `input1` формы отправим данные (`value` элемента `input1`) на сервер:

```
<input id=input1 name=input1 value=10 onchange="f_ajax1(this.value) ">
```

Листинг 2.2 содержит саму функцию.

Листинг 2.2

```
function SendRequest (arg)
{
// url
var url="file1.php";
```

```
//Назначаем пользовательский обработчик
Request.onreadystatechange = f_answer1;
// Инициализируем соединение
Request.open("post" ,url, true);
// отправляем заголовок
Request.setRequestHeader("Content-Type","application/x-www-form-urlencoded;
charset=utf-8");
//Посыпаем запрос
Request.send(arg);
}
```

Теперь разберемся с ответом сервера (листинг 2.3). Нужно знать только два момента:

- не делать ничего, пока свойство XMLHttpRequest.readyState не будет равно 4;
- сервер будет записывать свой ответ в свойстве XMLHttpRequest.responseText.

Листинг 2.3

```
function f_answer1 () {
    if (xmlHttp.readyState == 4) {
        var response = xmlHttp.responseText;
        alert(response);
    }
}
```

2.1.4. Варианты ответа от сервера

От сервера можно получить данные нескольких видов:

- обычный текст;
- XML;
- JSON.

Если вы получаете обычный текст, то можете сразу же направить его в контейнер, т. е. на вывод. При получении данных в виде XML вы должны обработать данные с помощью DOM-функций и представить результат с помощью HTML. JSON — это объектная нотация JavaScript. С ее помощью можно представить объект в виде строки (здесь можно привести аналогию с функцией сериализации). При получении JSON-данных вы должны выполнить их, чтобы получить полноценный объект JavaScript и произвести с ним необходимые операции.

Существуют множество AJAX-библиотек, упрощающих обмен данными между клиентом и сервером. Однако большинство этих решений полностью написаны на JavaScript и исполняются на машине клиента. Нередко разработчики на PHP плохо знают JavaScript, что делает очень затруднительным использование этих библиотек в PHP-приложениях. К счастью, есть фреймворк хаях, полностью написанный на PHP и включающий AJAX-функции.

2.2. Фреймворк хајах

Фреймворк хајах — это библиотека классов PHP с открытым исходным кодом, которая позволяет вам легко создавать мощные, Web-ориентированные AJAX-приложения, использующие HTML, CSS, JavaScript и PHP. Приложения, разработанные при помощи библиотеки хајах, могут асинхронно вызывать расположенные на сервере PHP функции и обновлять содержание без перезагрузки страницы. хајах предоставляет простую реализацию технологии AJAX, а, начиная с версии 0.5, еще и PHP-инструменты для формирования HTML-форм и документов. В отличие от многих других подобных библиотек, хајах позволяет разрабатывать AJAX-приложения, не требуя от разработчика знания JavaScript. Библиотека хајах распространяется по лицензии GNU Lesser General Public License (LGPL) и может быть использована для написания платного программного обеспечения. Сайт проекта <http://hajaxproject.org>. На момент написания книги доступна версия 0.5.

2.2.1. Как работает хајах

Библиотека хајах создает функции JavaScript в виде оболочки PHP-функций, которые вы можете вызывать с сервера из вашего приложения. Когда вызывается функция JavaScript, то она, являясь оболочкой для функции PHP, использует объект XMLHttpRequest для асинхронного соединения с объектом хајах на сервере, который вызывает соответствующую функцию PHP. После завершения этого действия, возвращается хајах XML-ответ от вызванной PHP-функции. Возвращенный XML-код содержит инструкции и данные, которые будут проанализированы специальными функциями JavaScript в хајах и использованы для обновления содержания вашего приложения.

2.2.2. Возможности хајах

Фреймворк хајах предлагает следующие возможности, которые вместе делают его уникальным и мощным инструментом:

- Может анализировать возвращенный код XML и автоматически его обрабатывать согласно инструкциям, находящимся в этом ответе. Поскольку хајах обрабатывает все это, то вам не нужно писать отдельные функции на JavaScript для того, чтобы обрабатывать возвращенный XML.
- Ориентирован на создание отношений между программным кодом и данными для хранения кода хајах отдельно от другого программного кода. Так как это объектно-ориентированный код, то вы всегда можете добавлять свои функции в класс `hajaxResponse`, используя метод `script()`.
- Работает в Firefox, Mozilla, Internet Explorer и Safari. Помимо обновления значений элементов (имеется в виду DOM) и `innerHTML`, хајах позволяет обновлять стили, классы CSS, значения переключателей и выпадающих списков или каких-либо других свойств элемента.
- Может работать с одно- и многоразмерными массивами, а также ассоциативными массивами из JavaScript в PHP как параметрами ваших функций хајах. В дополне-

ние, если вы вводите объект JavaScript в функцию хајах, функция PHP будет получать ассоциативный массив, определяющий свойства этого объекта.

- Предоставляет легкую асинхронную обработку форм. Используя метод `javascript_xajax.getFormValues()`, вы можете легко отправить массив данных в форме как параметры для асинхронной функции хајах: `xajax_processForm(xajax.getFormValues('formId'))`.

Если действие совершилось совместно с элементом формы `input` под именем `"checkbox[][]"` и `"name[first]"`, то создаются многоразмерные и ассоциативные массивы, такие как если бы вы отправляли форму через массив `$_GET`. С помощью хајах вы можете динамически подгружать дополнительный JavaScript-код для вашего приложения, чтобы при его исполнении менялись свойства элемента DOM.

- Хајах автоматически сравнивает данные, возвращенные из PHP-функций с текущими значениями свойства элемента, который вы хотите изменить. Свойство изменяется только в том случае, если это изменение актуально на данный момент. Это позволяет устраниТЬ мерцание, которое происходит, если элемент обновляется каждый раз через определенные промежутки времени. Каждая функция зарегистрируется для того, чтобы быть доступной через хајах, который имеет различные типы запросов. Все функции по умолчанию используют POST-запросы к серверу, за малым исключением — GET-запросы. Это сделано для большей безопасности запросов.
- Если не определен запрашиваемый URI, хајах пытается автоматически определить запрашиваемый URL скрипта. Алгоритм автоопределения хајах достаточно универсален, так что он будет работать как на безопасном протоколе HTTPS, так и на HTTP и на нестандартных портах.
- Хајах перекодирует все свои запросы и ответы в кодировку UTF-8, таким образом он поддерживает большой спектр различных знаков и языков.
- Хајах был протестирован на различных языках, включая испанский, русский и арабский. Почти весь JavaScript динамически подгружается через JavaScript-расширения.
- В шаблонном движке Smarty для создания переменной должен быть такой код: `$smarty->assign('xajax_javascript', $xajax->getJavascript())`. Когда используете хајах, подставляйте в заголовок тег `{$xajax_javascript}`.

Таким образом, хајах позволяет вам легко создавать мощные, Web-ориентированные AJAX-приложения на основе HTML, CSS, JavaScript и PHP. Приложения, разработанные при помощи библиотеки хајах, могут асинхронно вызывать расположенные на сервере PHP-функции и обновлять содержание без перезагрузки страницы.

2.2.3. Подключение хајах

Хајах — это PHP-библиотека, которая отличается тем, что позволяет выполнять JavaScript на основе PHP-кода. Весь процесс состоит из двух PHP-классов и обработчика XML на JavaScript. В общем, на PHP сначала инициализируется объект

и объявляются функции, которые будут отвечать на AJAX-запрос. В этих функциях необходимо использовать объект, который и будет генерировать XML-ответ. Для скачивания библиотеки хаах заходим по адресу <http://xajaxproject.org/en/download/> и нажимаем на ссылку **xajax 0.5 standard**. Скачиваем архив с библиотекой на компьютер (рис. 2.1).

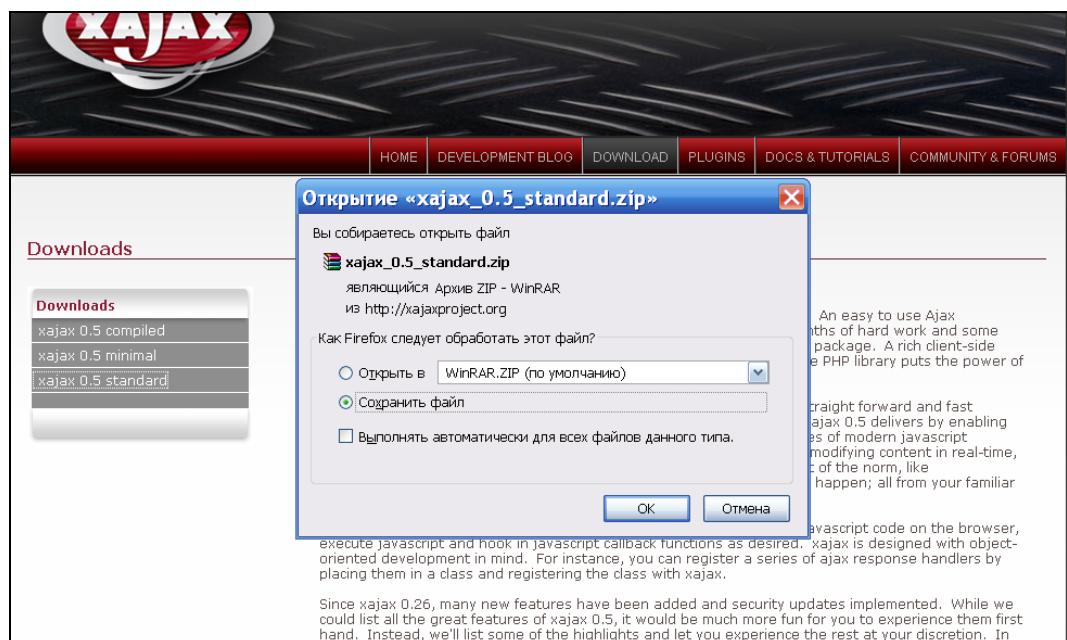


Рис. 2.1. Страница скачивания библиотеки хаах

Распакуем архив в корневой каталог сайта. Листинг 2.4 содержит код, который необходимо внести в файл для подключения хаах.

Листинг 2.4

```
<html>
<head>
<?php
    // подключение библиотеки
    require_once ("xajax_core/xajax.inc.php");
    // создать новый хаах-объект
    $xajax = new xajax();
    // регистрация функций
    $xajax->register(XAJAX_FUNCTION, "Function1");
    // разрешаем обрабатывать хаах асинхронные запросы
    $xajax->processRequest();
```

```
function Function1()
{
    $objResponse = new xajaxResponse();
    // код
    return $objResponse;
}

?>
<?php
echo $xajax->getJavascript( "" );
?>
</head>
<body>

</body>
</html>
```

Ясно видно, что объект `хаях` основной, в него регистрируются функции обработки, а `хаяхResponse` — вспомогательный, который генерирует XML, который потом распознается на уровне JavaScript и выполняет соответствующие действия. При вызове хаях-функций из JavaScript добавляется префикс `хаях_`:

```
<a 'href=javascript:void();' onclick='хаях_Function1();'></a>
```

2.2.4. Методы объекта `xajaxResponse`

Объект `хаяхResponse` имеет следующие методы: `assign`; `append`; `prepend`; `replace`; `remove`; `create`; `insert`; `insertAfter`; `clear`; `createInput`; `insertInput`; `insertInputAfter`; `removeHandler`; `includeScript`; `script`; `addEvent`; `call`; `alert`; `redirect`.

□ Рассмотрим подробнее каждый из этих методов.

ЗАМЕЧАНИЕ

На прилагаемом компакт-диске представлен Web-сайт — тренажер для изучения этих методов с просмотром результатов в визуальном режиме. Подробно работа тренажера будет рассмотрена в разд. 2.2.5.

2.2.4.1. Метод `assign`

Изменяет параметры HTML-элементов, таких как `innerHTML`, `style` и т. п.

`$objResponse->assign($sTarget, $sAttribute, $sData)` — заменяет значение `$sAttribute` элемента `$sTarget` на `$sData`.

Примеры

```
// установить новое содержимое элемента с id=div1 — <b>New</b>
$objResponse->assign("div1","innerHTML","<b>New</b>");
```

```
// установить цвет текста красный в элементе с id=div1
$objResponse->assign("div1","style.color","red");
// скрыть элемент с id=div1
$objResponse->assign("div1","style.display","block");
```

2.2.4.2. Метод *append*

Изменяет параметры элементов, добавляя данные в конец.

`$objResponse->append($sTarget, $sAttribute, $sData)` — добавляет `$sData` в конец значения атрибута `$sAttribute` элемента `$sTarget`.

Примеры

```
// добавить в конец содержимого элемента с id=div1 код HTML -
// <U>element</U>
$objResponse->append("div1","innerHTML","<U>element</U>");
// если HTML-код был <b>New</b>, то теперь будет -
// <b>New</b><U>element</U>
```

2.2.4.3. Метод *prepend*

Добавляет данные в начало.

`$objResponse->prepend($sTarget, $sAttribute, $sData)` — добавляет `$sData` в начало значения атрибута `$sAttribute` элемента `$sTarget`.

Примеры

```
// добавить в начало содержимого элемента с id=div1 код HTML -
// <U>element</U>
$objResponse->append("div1","innerHTML","<U>element</U>");
// если HTML-код был <b>New</b>, то теперь будет -
// <U>element</U><b>New</b>
```

2.2.4.4. Метод *replace*

Заменяет в элементе одни на другие, как `str_replace`.

`$objResponse->replace($sTarget, $sAttribute, $sSearch, $sData)` — заменяет найденное по маске `$sSearch` значение атрибута `$sAttribute` элемента `$sTarget` на `$sData`.

Примеры

```
// заменить в содержимом элемента с id=div1 код HTML -
// 1 на код 2<br>
$objResponse->replace("div1","innerHTML","1","2<br>");
// если HTML-код был <b>1234512345</b>, то теперь будет -
// <b>2<br>23452<br>2345</b>
// напоминает php функцию str_replace
```

2.2.4.5. Метод *remove*

Удаляет элемент.

`$objResponse->remove ($sTarget)` — уничтожает элемент `$sTarget`.

Пример

```
// удалить элемент с id=div1  
$objResponse->remove ("div1");
```

2.2.4.6. Метод *create*

Создает элемент.

`$objResponse->create ($sParent, $sTag, $sId, $sType = "")` — создает элемент `$sTag` с `id = $sId`, как потомок от `$sParent` с типом `$sType`.

Примеры

```
// создать элемент span с id=span1 в элементе с id=div1  
$objResponse->create ("div1","span","span1");  
// если на странице был элемент div с id=div1  
// <div id=div1></div>, то теперь будет  
// <div id=div1><span id=span1></span></div>
```

2.2.4.7. Метод *insert*

Вставляет новый элемент.

`$objResponse->insert ($sBefore, $sTag, $sId)` — вставляет элемент `$sTag` с ID — `$sId` до элемента `$sBefore`.

Примеры

```
// создать элемент span с id=span1 до элемента с id=div1  
$objResponse->create ("div2","span","span1");  
// если на странице был элемент div с id=div1  
// а внутри него элемент div с id=div2  
// <div id=div1><div id=div2></div></div>, то теперь будет  
// <div id=div1><span id=span1></span><div id=div2></div></div>
```

2.2.4.8. Метод *insertAfter*

Добавляет элемент после заданного элемента.

`$objResponse->insertAfter ($sAfter, $sTag, $sId)` — вставляет элемент `$sTag` с ID — `$sId` после элемента `$sAfter`.

Примеры

```
// создать элемент span с id=span1 после элемента с id=div1  
$objResponse->create ("div2","span","span1");
```

```
// если на странице был элемент div с id=div1
// а внутри него элемент div с id=div2
// <div id=div1><div id=div2></div></div> , то теперь будет
// <div id=div1><div id=div2><span id=span1></span></div></div>
```

2.2.4.9. Метод *clear*

Очищает содержимое элемента.

`$objResponse->clear($sTarget, $sAttribute)` — очищает значение атрибута `$sAttribute` элемента `$sTarget`.

Примеры

```
// очищает содержимое элемента с id=div1
$objResponse->clear("div2","innerHTML");
// очищает содержимое свойства color элемента с id=div1
$objResponse->clear("div2","color");
```

2.2.4.10. Метод *createInput*

Создает элемент формы.

`$objResponse->createInput($sParent, $sType, $sName, $sId)` — создает элемент HTML-формы как дочерний элемент от элемента `$sParent`, с типом `$sType`, именем `$sName` и ID — `$sId`.

Пример

```
// создает в форме с id=form1 элемент input
// с id=input2 и name=input2
$objResponse->createInput("form1","input","input2","input2");
```

2.2.4.11. Метод *insertInput*

Создает элемент формы.

`$objResponse->insertInput($sBefore, $sType, $sName, $sId)` — создает элемент HTML-формы до элемента `$sBefore`, с типом `$sType`, именем `$sName` и ID — `$sId`.

Примеры

```
// создает в форме элемент input
// с id=input2 и name=input2
// до элемента с id=input1
$objResponse->insertInput("input1","input","input2","input2");
```

2.2.4.12. Метод *insertInputAfter*

Создает элемент формы.

`$objResponse->insertInputAfter($sAfter, $sType, $sName, $sId)` — создает элемент HTML-формы после элемента `$sAfter`, с типом `$sType`, именем `$sName` и ID — `$sId`.

Пример

```
// создает в форме элемент input  
// с id=input2 и name=input2  
// после элемента с id=input1  
$objResponse->insertInputAfter("input1","input","input2","input2");
```

2.2.4.13. Метод *removeHandler*

Удаление функции обработки событий.

`$objResponse->removeHandler($sTarget, $sEvent, $sHandler)` — удаляет js-функцию `$sHandler` для обработки события `$sEvent` для элемента `$sTarget`.

Пример

```
// удаляет функцию fun_over для события onmouseover  
// для элемента с id=div1  
$objResponse->removeHandler("div1","onmouseover","fun_over");
```

2.2.4.14. Метод *includeScript*

Подключает внешний js-файл.

`$objResponse->includeScript($sPath)` — подключает внешний js-файл, путь к которому `$sPath`.

Пример

```
// подключает внешний js-файл  
// чтобы, допустим, первая страница загружалась не слишком долго  
// внешний js-файл загружаем не в начале, а только тогда,  
// когда он будет использоваться  
$objResponse->includeScript("js/jquery.fancybox-1.3.0.js");
```

2.2.4.15. Метод *script*

Добавляет прописанную вручную js-обработку.

`$objResponse->script($sScript)` — выполняет код JavaScript, содержащийся в `$sScript`.

Примеры

```
// выполнить js - выдать окно alert с сообщением Сообщение!!!  
$objResponse->script("alert('Сообщение!!!!')");  
// элемент с id=d1 в зону видимости страницы  
$objResponse->script("document.getElementById('d1').scrollIntoView();");  
// установить цвет текста красный в элементе с id=div1  
// (аналогично $objResponse->assign("d1","style.color","red");)  
$objResponse->script("document.getElementById('d1').style.color='red';");
```

2.2.4.16. Метод *addEvent*

Создает новый объект event.

`$objResponse->event($sTarget, $sEvent, $sScript)` — создает событие `$sEvent`, привязывая его к элементу `$sTarget` и связывая с ним код `$sScript`.

Примеры

```
// создаем новые события для элемента с id=div1
// onmouseover - при наведении мыши на объект цвет элемента красный
// onmouseout - при наведении мыши на объект цвет элемента синий
$objResponse->event("div1","onmouseover","this.style.color='red'");
$objResponse->event("div1","onmouseout","this.style.color='blue'");
```

2.2.4.17. Метод *call*

Вызывает заданную js-функцию с заданными параметрами.

`$objResponse->call($sFunc, $args, ...)` — вызывает js-функцию `$sFunc` с заданными параметрами `$args`.

Пример

```
// вызывает js-функцию с тремя аргументами my_function("div1",4,10);
$objResponse->call("my_function", "div1", 4, 10))
```

2.2.4.18. Метод *alert*

Создает оповещение.

`$objResponse->alert($sMsg)` — окно-предупреждение JavaScript с текстом `$sMsg`.

Примеры

```
// выдать окно alert с сообщением Сообщение!!!
$objResponse->script("alert('Сообщение!!!');");
```

2.2.4.19. Метод *redirect*

Создает перенаправление на другую страницу, возможно, через некоторое время.

`$objResponse->redirect($sURL)` — создает перенаправление на страницу `$sURL`.

Примеры

```
// перенаправляет на другую страницу
$objResponse->redirect("http://www.site.ru/register.php");
```

2.2.5. Сайт — тренировочный стенд для изучения хаах

Для лучшего усвоения материала я создал небольшое Web-приложение для изучения response-методов библиотеки хаах. Сайт позволяет изучать асинхронно вызываемые функции хаах (response-методы) в режиме тренировочного стендса — выбираешь параметры и сразу же видишь результат на странице. Сайт сделан с использованием библиотеки хаах без перезагрузки страницы. Проект находится на прилагаемом компакт-диске в папке book_primers/1 и после установки на ваш компьютер и запуска программной оболочки "Денвер" (см. разд. 1.6) запускается из браузера по адресу http://book_primers/1. Наберите в адресной строке сайта http://book_primers/1 и увидите страницу, изображенную на рис. 2.2.

Для удобства восприятия методы объекта `ajaxResponse` разбиты на три группы:

1. Работа с элементами:

- `assign` — изменяет параметры HTML-элементов, таких как `innerHTML`, `style` и т. п.;
- `append` — также изменяет параметры элементов, добавляя данные в конец;
- `prepend` — добавляет данные в начало;
- `replace` — как вы догадались, заменяет в элементе одни фрагменты данных на другие, как PHP-функция `str_replace`;
- `remove` — удаляет элемент;
- `create` — создает элемент;
- `insert` — вставляет новый элемент;
- `insertAfter` — добавляет после заданного элемента;
- `clear` — очищает содержимое элемента.

2. Работа с полями ввода:

- `createInput` — создает поле;
- `insertInput` — добавляет поле до переданного в аргументе;
- `insertInputAfter` — добавляет поле после переданного в аргументе.

3. Особые процессы:

- `removeHandler` — удаляет функцию для обработки события;
- `includeScript` — подключает внешний js-файл;
- `script` — добавляет прописанную вручную js-обработку;
- `addEvent` — создает новый объект `event`;
- `call` — вызывает заданную js-функцию с заданными параметрами;
- `alert` — создает оповещение;
- `redirect` — создает перенаправление на другую страницу, возможно, через некоторое время.

Как работать с сайтом? При нажатии на ссылку **Работа с элементами** появляется страница, изложенная на рис. 2.3.

Самые популярные Начальная страница Последние заголовки Каталоги английский Russian

localhost >> localhost >> book... Пособие по изучению хајајк книге

Xajax

```
хайајк – это php библиотека, которая отличается тем, что позволяет выполнять javascript на основе php-кода. Весь процесс состоит из двух php классов и обработчика xml на javascript. В общем – на php сначала инициализируется объект и объявляются функции, которые будут отвечать на ajax-запрос. В этих функциях необходимо использовать объект, который будет генерировать xml-ответ.
```

```
require_once ("xajax_core/xajax.inc.php");
$xajax = new xajax('index.server.php');
$xajax->register(XAJAX_FUNCTION,"F1");
$xajax->processRequest();
function F1()
{
$ObjResponse = new xajaxResponse();
$ObjResponse->assign("data","innerHTML",$content);
return $ObjResponse;
}
```

Ясно видно что хајајк объект основной, в него регистрируются функции обработки, а xajaxResponse вспомогательный, который генерирует XML, который потом распознается на уровне javascript и выполняет соответствующие действия.

Возможности – Класс xajaxResponse имеет следующие наиболее используемые методы:

- * Работу с элементами:
 - o assign изменяет параметры html-элементов, будь то innerHTML, style и тп.
 - o append также изменяет параметры элементов, добавляя в конец данные
 - o prepend добавляет данные в начало
 - o replace как вы догадались – заменяет в элементе одни на другие, как str_replace
 - o remove удаляет элемент
 - o create создаёт элемент
 - o insert вставляет новый элемент

Готово

Рис. 2.2. Страница сайта — тренировочного стенда

Методы xajaxResponse для работы с элементами

- o **assign**
изменяет параметры html-элементов, будь то innerHTML, style и тп.
\$ObjResponse->assign(\$sTarget, \$sAttribute, \$sData);
- заменяет значение \$sAttribute элемента \$sTarget на \$sData.
- o **append**
изменяет параметры элементов, добавляя в конец данные
\$ObjResponse->append(\$sTarget, \$sAttribute, \$sData);
- добавляет \$sData в конец значения атрибута \$sAttribute элемента \$sTarget
- o **prepend**
добавляет данные в начало
\$ObjResponse->prepend(\$sTarget, \$sAttribute, \$sData);
- добавляет \$sData в начало значения атрибута \$sAttribute элемента \$sTarget.
- o **replace**
как вы догадались – заменяет в элементе одни на другие, как str_replace
\$ObjResponse->replace(\$sTarget, \$sAttribute, \$sSearch, \$sData);
- заменяет найденное по маске \$sSearch значение атрибута \$sAttribute элемента \$sTarget на \$sData.
- o **remove**
удаляет элемент
\$ObjResponse->remove(\$sTarget);
- уничтожает элемент \$sTarget
- o **create**
создаёт элемент
\$ObjResponse->create(\$sParent, \$sTag, \$sId, \$sType = "");
- создает элемент \$sTag с id - \$sId, как потомок от \$sParent и типом \$sType.
Тип удобно использовать для создания элементов форм
- o **insert**
вставляет новый элемент
\$ObjResponse->insert(\$sBefore, \$sTag, \$sId);
- вставляет элемент \$sTag с id - \$sId до элемента \$sBefore.
- o **insertAfter**
добавляет после заданного элемента
\$ObjResponse->insertAfter(\$sAfter, \$sTag, \$sId);
javascript:void()

Рис. 2.3. Выбор методов для работы с элементами

Далее выбираем какой-нибудь метод, например `assign`, и попадаем на страницу, изображенную на рис. 2.4. Выбираем в форме значения атрибутов и нажимаем на кнопку **Выполнить**.

Результат работы функции показан на рис. 2.5: изменился фон элемента `span13`. В блоке **Код** видим код отправленной команды.

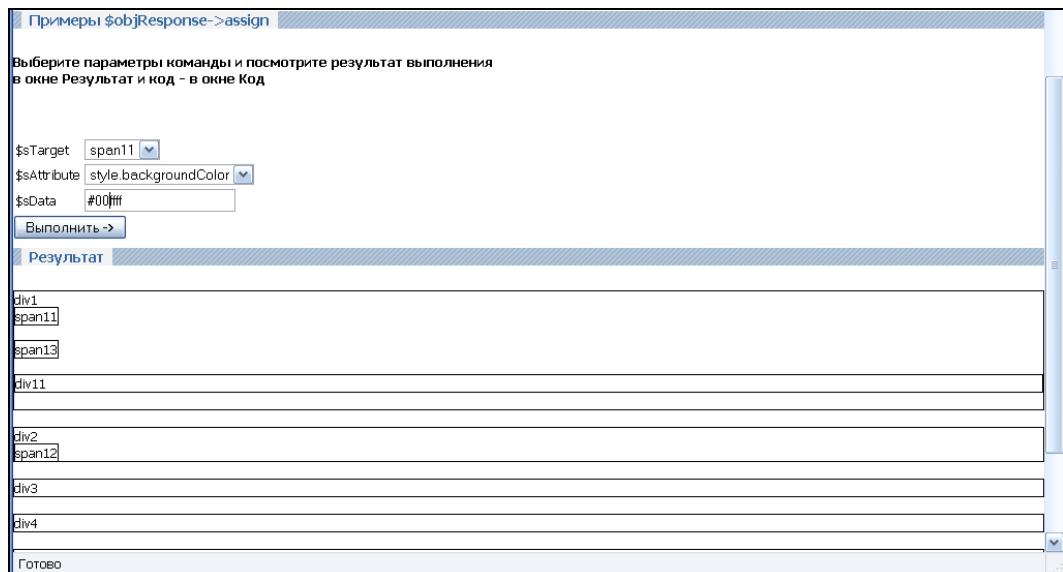


Рис. 2.4. Выбор параметров функции `assign`

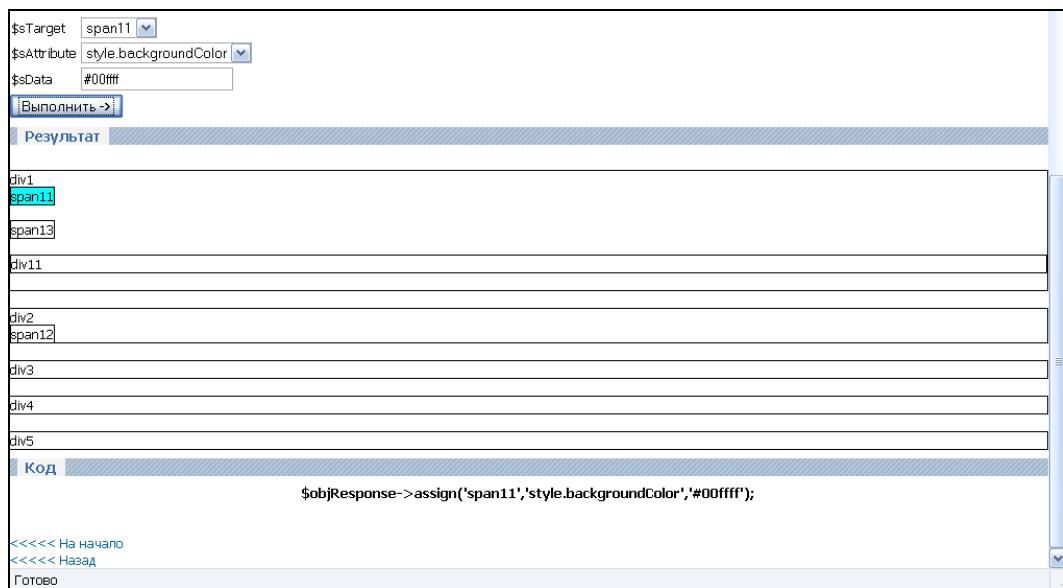


Рис. 2.5. Вид страницы после выполнения функции `assign`

Теперь вернемся по ссылке **Назад**, выберем метод `remove` (рис. 2.6) и укажем в выпадающем списке какой-нибудь элемент, например `span13`.

Нажимаем на кнопку **Выполнить** и на рис. 2.7 видим результат выполнения функции: элемент `span13` удален.

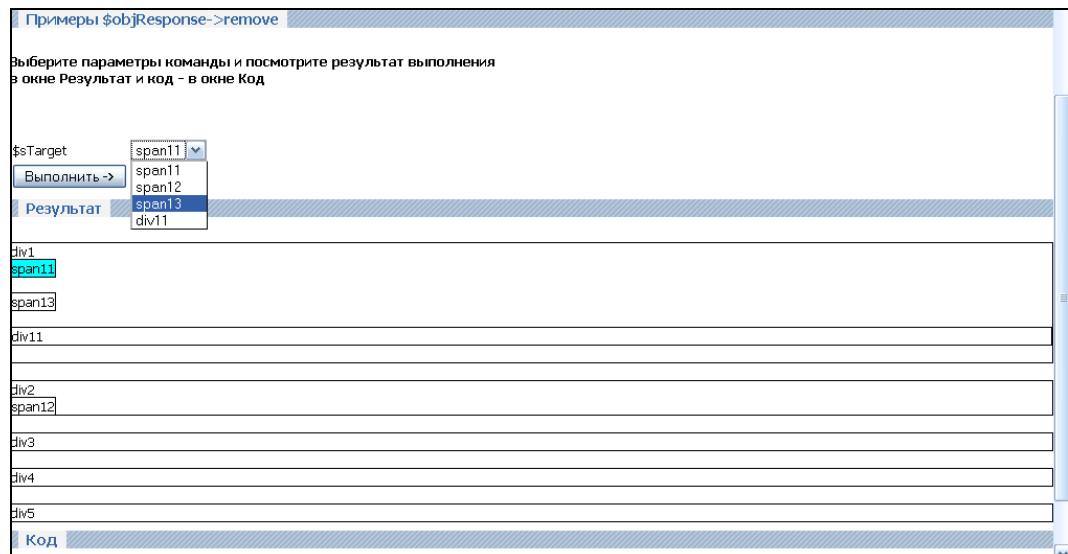


Рис. 2.6. Выбор параметров функции `remove`

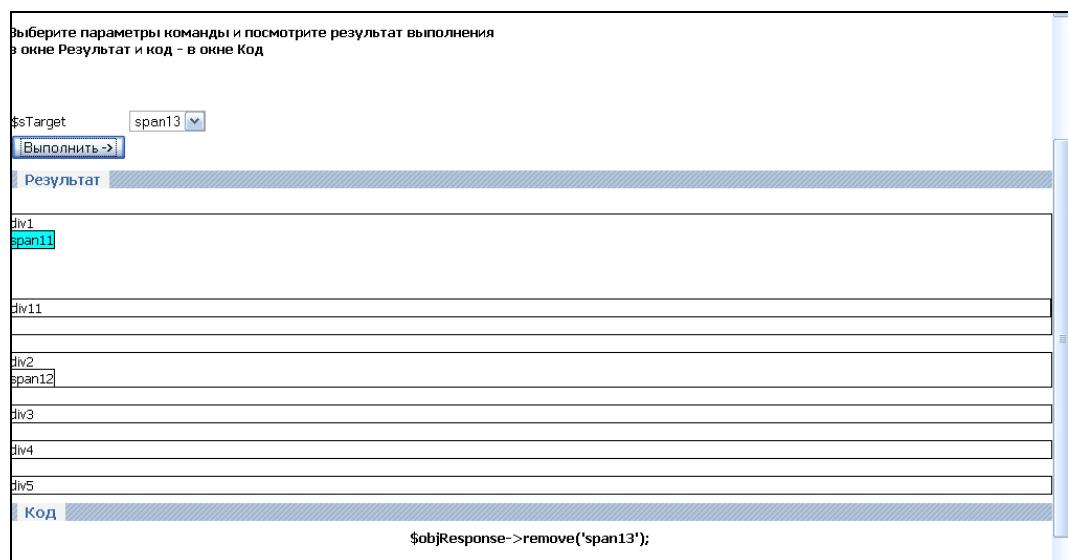


Рис. 2.7. Вид страницы после выполнения функции `remove`

Я думаю, принцип работы со стендом понятен. Надеюсь, он поможет вам лучше изучить работу асинхронных функций библиотеки хаах. On-line-версия тренировочного стенда размещена в Интернете по адресу <http://goodtovars.ru/stend>.

ЗАМЕЧАНИЕ

При создании, удалении блоков вся информация сохраняется в базе данных, поэтому вид блока **Результат** может отличаться от приведенного ранее. С другой стороны, стенд не позволяет удалить все блоки, и блок **Результат** никогда не будет пустым.

2.2.6. Глобальные переменные хаах

Знание возможностей хаах состоит не только в умении написать страницу, которая будет работать без перезагрузки, но и в способности настраивать и менять те многочисленные параметры, которые могут кардинально изменить работу хаах.

2.2.6.1. Глобальные константы

Глобальные константы объекта хаах:

- XAJAX_DEFAULT_CHAR_ENCODING string (по умолчанию "utf-8") — используется как в классах хаах, так и xajaxResponse. Вы можете сами задать значение этой константы;
- XAJAX_GET int (по умолчанию 0) — показывает, что задан метод GET HTTP-запроса, используемый в хаах;
- XAJAX_POST int (по умолчанию 1) — в хаах установлен метод POST HTTP-запроса.

2.2.6.2. Методы объекта хаах

Объект хаах имеет следующие методы:

- xajax (string \$sRequestURL="", string \$sWrapperPrefix="xajax_", string \$sEncoding=XAJAX_DEFAULT_CHAR_ENCODING, boolean \$bDebug=false) — конструктор. Вы можете сразу установить нужные вам значения в конструкторе или же после посредством методов.

Параметры:

- \$sRequestURL — тип string (по умолчанию текущий URL браузера);
- \$sWrapperPrefix — тип string (по умолчанию "хаах_");
- \$sEncoding — тип string (по умолчанию равно глобальной константе);
- \$bDebug Mode — тип boolean (по умолчанию false).

- setRequestURI (string \$sRequestURL) — устанавливает URI, к которому будет сделан запрос.

Параметры:

- \$sRequestURL — это URL (может быть абсолютным или относительным) в PHP-скрипте, который должен быть запрошен объектом хаах.

Пример:

```
$xajax->setRequestURL("http://www.site.ru");
```

- `debugOn ()` — установить вывод отладочных сообщений (JavaScript alerts) для запроса хаях.
- `debugOff ()` — скрыть вывод отладочных сообщений (JavaScript alerts) для запроса хаях.
- `statusMessagesOn ()` — включает вывод сообщений в строку состояния браузера хаях.
- `statusMessagesOff ()` — отключает вывод сообщений в строку состояния браузера хаях (установлено по умолчанию).
- `waitCursorOn ()` — включает ожидание показа курсора браузером (установлено по умолчанию).
- `waitCursorOff ()` — отключает ожидание показа курсора браузером.
- `exitAllowedOn ()` — включает прерывание процесса хаях после того, как запрос был выслан и ответ был получен браузером (установлено по умолчанию).
- `exitAllowedOff ()` — отключает прерывание процесса хаях после того, как запрос был выслан и ответ был получен браузером.
- `errorHandlerOn ()` — включает обработку ошибок системой после того, как произошла ошибка в ходе приема запроса и отправки ответа сервером (JavaScript alerts).
- `errorHandlerOff ()` — отключает обработку полученных ошибок в хаях (установлено по умолчанию).
- `setLogFile (string $sFilename)` — определяет лог-файл, в который будет записана ошибка при исполнении запроса хаях. Если вы не вызываете этот метод, то лог-файл писаться не будет.

Применение:

```
$xajax->setLogFile("/xajax_logs/errors.log");
```

- `setWrapperPrefix ($sPrefix)` — устанавливает префикс, который будет добавлен к функциям JavaScript (по умолчанию "хаях_").
- `setCharEncoding (string $sEncoding)` — устанавливает кодировку для вывода HTML, основанный на переменной `$sEncoding`, которая является строкой, содержащей набор символов, определяющих кодировку. Обычно вы не нуждаетесь в применении этого метода, т. к. кодировка устанавливается автоматически, на основе глобальной константы `XAJAX_DEFAULT_CHAR_ENCODING`.

Пример:

```
$xajax->setCharEncoding("windows-1251");
```

//или:

```
$xajax->setCharEncoding("utf-8");
```

- `registerFunction (mixed $mFunction, string $sRequestType=XAJAX_POST)` — регистрация PHP-функции или метода для того, чтобы она могла быть вызвана через хаях в вашем коде JavaScript. Если вы хотите зарегистрировать статический метод, введите следующий массив — `array("myFunctionName", "myClass", "myMethod")`.

ЗАМЕЧАНИЕ

Для метода экземпляров класса используйте объектную переменную в качестве второго аргумента (В PHP 4 проверьте, что перед переменной поставлен знак &). Функцию, которую вы вызываете через JavaScript, называйте так, чтобы она имела уникальное имя и не конфликтовала с другими функциями.

Параметры:

- \$mFunction — строка, содержащая имя функции или вызываемый массив объектов;
- \$sRequestType — тип запроса (XAJAX_GET/XAJAX_POST), который будет использоваться функцией (по умолчанию XAJAX_POST).

Пример:

```
$xajax->registerFunction("myFunction");  
//или:  
$xajax->registerFunction(array("myFunctionName", &$myObject,  
    "myMethod"));
```

- registerExternalFunction (mixed \$mFunction, string \$sIncludeFile, string \$sRequestType=XAJAX_POST) — зарегистрировать PHP-функцию, которая расположена в другом файле и может вызываться из хаях. Если функция запрошена, внешний файл будет включен в описание, перед тем как будет вызвана функция.

Параметры:

- \$mFunction — строка, содержащая имя функции или вызываемый массив объектов (см. registerFunction() для более полной информации по вызываемому массиву объектов);
- \$sIncludeFile — строка, содержащая путь и имя включаемого файла;
- \$sRequestType — тип запроса (XAJAX_GET/XAJAX_POST), использованный для этой функции. По умолчанию XAJAX_POST.

Пример:

```
$xajax->registerExternalFunction("myFunction", "myFunction.inc.php",  
    XAJAX_POST);
```

- registerCatchAllFunction (mixed \$mFunction) — регистрирует функцию PHP, которая будет вызвана, если хаях не может найти функцию, вызываемую посредством JavaScript. Метод действует только при вызове метода JavaScript xajax.call() напрямую. Используйте эту возможность для обработки неописанных ситуаций. \$mFunction — строка, содержащая имя функции или вызываемый массив объектов (см. registerFunction() для более полной информации по вызываемому массиву объектов).

Пример:

```
$xajax->registerCatchAllFunction("myCatchAllFunction");
```

- registerPreFunction (mixed \$mFunction) — регистрирует функцию PHP для исполнения, перед тем как хаях выполнит запрашиваемую функцию. Хаях

будет автоматически добавлять запрос функции на возвращенный ответ "предфункции" для того, чтобы собрать из них один ответ. Другая возможность — не ждать ответа, но тогда будет возвращаться массив, первый элемент которого есть `false` (`boolean`), а второй является ответом. В этом случае ответ от "предфункции" будет возвращен в браузер без вызова функции, запрошенной хаах.

Параметры:

- `$mFunction` — строка, содержащая имя функции или вызываемый массив объектов (см. `registerFunction()` для более полной информации по вызываемому массиву объектов).

Пример:

```
xaajax->registerPreFunction("myPreFunction").
```

- `canProcessRequests ()` — возвращает `true`, если хаах может обработать запрос, иначе `false`. Позволяет узнать, прошел запрос или нет.
- `getRequestMode ()` — возвращает текущий вид запроса (`XAJAX_GET` или `XAJAX_POST`) или `-1`, если не определен.
- `processRequests ()` — это основной движок для коммуникации в хаах, который обрабатывает все входящие запросы хаах, вызывает соответствующие PHP-функции (или методы `class/object`) и выводит XML-ответы назад в обработчик ответов JavaScript. Если ваш запрашиваемый URI отличается от URI вашей страницы, то эта функция должна быть вызвана перед выводом заголовков или HTML.
- `printJavascript (string $sJsURI="", string $sJsFile=NULL, string $sJsFullFilename=NULL)` — хаах печатает JavaScript-код в вашу страницу, который является результатом вывода метода `getJavascript()`. Метод вызывается только между тегами в вашей HTML-странице.

Параметры:

- `$sJsURI` — относительный адрес папки, где установлен хаах. Для PHP-файла, находящегося в каталоге `http://www.myserver.com/myfolder/mypage.php` и хаах, который был установлен в каталог `http://www.myserver.com/anotherfolder`, по умолчанию предполагается, что хаах находится в той же папке, что и ваш PHP-файл;
 - `$sJsFile` — относительный путь к папке/файлам движка хаах JavaScript, расположенного в инсталляционной папке (по умолчанию `xaah_js/xaah.js`);
 - `$sJsFullFilename` — может быть установлен абсолютный путь к файлу `xaah.js`. Этот аргумент нужно использовать, только если вы переместили папку `xaah_js`.
- `getJavascript (string $sJsURI="", string $sJsFile=NULL, string $sJsFullFilename=NULL)` — возвращает код хаах JavaScript, который должен быть добавлен в вашу HTML-страницу между тегами. Аргументы функции такие же, как и в методе `printJavascript()`.

2.3. Примеры использования хаах

Приступим теперь к рассмотрению примеров использования хаах.

2.3.1. Форма регистрации с проверкой правильности заполнения полей "на лету"

Классический вариант применения технологии AJAX — проверка правильности заполнения полей формы до отправки данных на сервер. Немного усложним этот пример. Во-первых, блокируем возможность нажатия на кнопку `submit` до правильного заполнения всех полей формы и, во-вторых, отправим данные на сервер без перезагрузки страницы. Файлы примера расположены на прилагаемом компакт-диске в папке `book_primers/2-1`. В файле `index.php` подключаем файлы библиотеки хаах, создаем объект хаах, регистрируем функции, к которым можно обращаться из JavaScript (`хаях_Control`, `хаях_Result`), и разрешаем обрабатывать хаах асинхронные запросы. Это серверная часть файла `index.php`. Клиентская часть — HTML-код формы регистрации. Проверка полей формы происходит следующим образом. Для предотвращения отправки непроверенных данных изначально кнопка `submit` неактивна (свойство `disabled=true`). По событию `onchange` каждого элемента формы `Form1` происходит вызов хаах-функции `Control`, в которую передаются все данные формы (`хаях.getFormValues("Form1")`). Функция проверяет правильность заполнения каждого поля. При неверном заполнении выводится сообщение `ERROR`, при правильном — `OK`. Для предотвращения отправки непроверенных данных изначально кнопка `submit` неактивна (свойство `disabled=true`). В случае правильного заполнения всех полей, делаем кнопку `submit` активной. Содержимое файла `index.php` приведено в листинге 2.5.

Листинг 2.5

```
<?php
    // подключение библиотеки
    require_once ("хаях_core/хаях.inc.php");
    // подключение внешних файлов
    require_once ("control.php");
    require_once ("result.php");
    // создать новый хаах-объект
    $хаях = new хаях();
    // регистрация функций
    $хаях->register(XAJAX_FUNCTION, "Control");
    $хаях->register(XAJAX_FUNCTION, "Result");
    // разрешаем обрабатывать хаах асинхронные запросы
    $хаях->processRequest();

?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
<title> Пример 1 (глава 2) к книге "ХАЈАХ" </title>
<?php $xajax->printJavascript(''); ?>
</head>
<body>
<div id=header1><b>Примеры к книге "ХАЈАХ" (глава 2 пример 1)<br>
Форма регистрации с проверкой полей "на лету"</b></div>
<br>
<!-- Форма -->
<div id='div1'>
<form id='Form1' action='javascript:void(null);' onsubmit='
    xajax.$("Button_Form1").disabled=true;
    xajax.$("Button_Form1").value="Подождите...";
    xajax_Result(xajax.getFormValues("Form1"));'>
<table>
<!-- login -->
<tr>
<td>Логин (5-15 буквы, цифры)</td>
<td><input type='text' name='login' id='login' value=''
    onchange='xajax_Control(xajax.getFormValues("Form1"));'></td>
<td><div id='error_login'><font color='red'>no</font></div></td>
</tr>
<!-- password -->
<tr>
<td>Пароль (5-15 буквы, цифры)</td>
<td><input type='password' name='password' id='password'
    onchange='xajax_Control(xajax.getFormValues("Form1"));'></td>
<td><div id='error_password'><font color='red'>no</font></div></td>
</tr>
<!-- подтверждение password -->
<tr>
<td>Повторите пароль</td>
<td><input type='password' name='password1' id='password1' value=''
    onchange='xajax_Control(xajax.getFormValues("Form1"));'></td>
<td><div id='error_password1'><font color='red'>no</font></div></td>
</tr>
<!-- email -->
<tr>
<td>Ваш e-mail</td>
<td><input type='input' name='email' id='email' value=''
```

```

onchange='xajax_Control(xajax.getFormValues("Form1"));'></td>
<td><div id='error_email'><font color='red'>no</font></div></td>
</tr>
<!-- submit -->
<tr>
<td></td>
<td><input type='submit' id='Button_Form1' disabled=true
value='Зарегистрироваться ->' ></td>
<td></td>
</tr>
</table>
</div>
<!-- блок для вывода результата-->
<div id='result'></div>
</body>
</html>

```

Функция `Control()` находится в файле `control.php`. Проверку данных формы проводим, используя регулярные выражения. Если данные поля не соответствуют шаблону, в блок `error` для этого поля выводим сообщение об ошибке (рис. 2.8).

При правильном заполнении всех полей активируется кнопка `submit` (рис. 2.9). Содержимое файла `control.php` приведено в листинге 2.6.

Форма регистрации с проверкой полей "на лету"

Логин (5-15 буквы,цифры)	<input type="text" value="Вася"/>	ERROR
Пароль (5-15 буквы,цифры)	<input type="password" value="*****"/>	OK
Повторите пароль	<input type="password" value="*****"/>	OK
Ваш e-mail	<input type="text" value="my_mail.ru"/>	ERROR
<input type="button" value="Зарегистрироваться ->"/>		

Рис. 2.8. Результат проверки полей формы

Форма регистрации с проверкой полей "на лету"

Логин (5-15 буквы,цифры)	<input type="text" value="Василий"/>	OK
Пароль (5-15 буквы,цифры)	<input type="password" value="*****"/>	OK
Повторите пароль	<input type="password" value="*****"/>	OK
Ваш e-mail	<input type="text" value="my_mail@mail.ru"/>	OK
<input type="button" value="Зарегистрироваться ->"/>		

Рис. 2.9. Активизация кнопки отправки формы

Листинг 2.6

```
// Проверка правильности заполнения полей
// при регистрации пользователя
function Control($Id)
{
    $objResponse = new xajaxResponse();
    $count=0;
    // login
    if(!ereg("^[a-z,A-Z,а-я,А-Я,0-9]{5,15}$",$Id[login]))
        $objResponse->assign("error_login","innerHTML","<font
            color='red'>ERROR</font>");
    else
    {
        $objResponse->assign("error_login","innerHTML","<font
            color='blue'>OK</font>");
        $count++;
    }
    // password
    if(!ereg("^[a-z,A-Z,а-я,А-Я,0-9]{5,15}$",$Id[password]))
        $objResponse->assign("error_password","innerHTML","<font
            color='red'>ERROR</font>");
    else
    {
        $objResponse->assign("error_password","innerHTML","<font
            color='blue'>OK</font>");
        $count++;
    }
    // password1
    if($Id[password1]==$Id[password] && strlen($Id[password1])>0)
    {
        $objResponse->assign("error_password1","innerHTML","<font
            color='blue'>OK</font>");
        $count++;
    }
    elseif(strlen(trim($Id[password1]))==0)
    {
        $objResponse->assign("error_password1","innerHTML","<font
            color='red'>ERROR</font>");
    }
    else
    {
        $objResponse->assign("error_password1","innerHTML","<font
            color='red'><>password!</font>");
    }
}
```

```
    }
    // email
    if(ereg("^( [a-z,0-9,-_\.]{2,20}) ([\@]{1}) ([a-z,0-9,-_"
        {2,20}) ([\.] {1}) ([a-z,] {1,3}) $", $Id[email]))
    {
        $objResponse->assign("error_email","innerHTML","<font
            color='blue'>OK</font>");
        $count++;
    }
else
    $objResponse->assign("error_email","innerHTML","<font
        color='red'>ERR</font>");
// все поля правильно заполнены ?
// да
if($count==4)
    $objResponse->assign("Button_Form1","disabled",false);
// нет
else
    $objResponse->assign("Button_Form1","disabled",true);

return $objResponse;
}
```

Для отправки значений формы на сервер без перезагрузки страницы блокируем событие `action` формы. Значения будем отправлять по событию `onsubmit`. Данные передаются хайх-функции `Result`, которая находится в файле `result.php`. Функция формирует контент и выводит его в блок `result` (рис. 2.10). После этого поля формы обнуляются, и кнопка `submit` делается неактивной.

Форма регистрации с проверкой полей "на лету"

Логин (5-15 буквы,цифры)	<input type="text"/>	OK
Пароль (5-15 буквы,цифры)	<input type="text"/>	OK
Повторите пароль	<input type="text"/>	OK
Ваш e-mail	<input type="text"/>	OK
<input type="button" value="Зарегистрироваться ->"/>		

Успешная регистрация
Ваши данные:
Логин - Василий
Пароль - 123456
E-mail - my_mail@mail.ru

Рис. 2.10. Отправка значений формы и ответ сервера

Содержимое файла result.php приведено в листинге 2.7.

Листинг 2.7

```
function Result($Id)
{
    $objResponse = new xajaxResponse();
    // сделать кнопку неактивной
    $objResponse->assign("Button_Form1","disabled",true);
    $objResponse->assign("Button_Form1","value","Зарегистрироваться ->");
    // обнуление значений формы
    $objResponse->assign("login","value","");
    $objResponse->assign("password","value","");
    $objResponse->assign("password1","value","");
    $objResponse->assign("email","value","");
    // формирование контента
    $text1="Успешная регистрация<br>";
    $text1.="Ваши данные:<br>";
    $text1.="Логин – ".$Id[login]."<br>";
    $text1.="Пароль – ".$Id[password]."<br>";
    $text1.="E-mail – ".$Id[email]."<br>";
    // вывод контента
    $objResponse->assign("result","innerHTML",$text1);
    return $objResponse;
}
```

Внося необходимые изменения, можно использовать этот пример в своих проектах.

2.3.2. Динамически подгружаемые select-элементы

На многих сайтах вам наверняка приходилось видеть динамически подгружаемые многоуровневые элементы ввода, когда содержимое нижних динамически меняется в зависимости от выбора в вышестоящем элементе. Например, на сайте www.mail.ru так организован выбор населенного пункта (Регион → Район → Город).

В качестве примера, создадим подобный скрипт и мы. Файлы примера расположены на компакт-диске в папке book_primers/2-2. Нам понадобится готовая база населенных пунктов России. Вы думаете, что найти такую базу в Сети нереально? Отчасти это так. Но на сайте Главного научно-исследовательского вычислительного центра Федеральной налоговой службы (www.gnivc.ru) в свободном доступе находится КЛАДР (Классификатор адресов России). Он состоит из таких значений, как индекс, регион (субъект РФ), район, город, населенный пункт, улица, дом. Справочник представлен в формате XLS. Работу по переводу данных для региона

(субъекта РФ), района, города и населенного пункта я уже предварительно провел. Дамп базы данных размером 24 Мбайт находится на компакт-диске в папке book_primers/2-2/kladr_primer2-2.sql

ЗАМЕЧАНИЕ

Возможно при импорте дампа kladr.sql будет выдаваться ошибка. Для ее устранения зайдите в файл php.ini, расположенный в каталоге установки Денвера в папке usr/local/php5, и измените значение параметра upload_max_filesize=8M на большее, например, upload_max_filesize = 30M. Не забудьте перезагрузить Денвер.

Структура таблицы primer_kladr:

- id — первичный ключ;
- id_region — ID региона;
- id_rayon — ID района;
- id_town — ID города;
- id_punkt — ID населенного пункта;
- name — название объекта;
- socr — наименование типа (край, область, хутор и пр.);
- zindex — почтовый индекс.

Дамп для создания структуры таблицы primer_kladr приведен в листинге 2.8.

Листинг 2.8

```
CREATE TABLE 'book_xajax1'.'primer_kladr' (
'id' bigint( 12 ) NOT NULL AUTO_INCREMENT ,
'id_region' int( 2 ) NOT NULL default '0',
'id_rayon' int( 3 ) NOT NULL default '0',
'id_town' int( 3 ) NOT NULL default '0',
'id_punkt' int( 3 ) NOT NULL default '0',
'name' varchar( 100 ) CHARACTER SET cp1251 default NULL ,
'socr' varchar( 10 ) CHARACTER SET cp1251 default NULL ,
'zindex' int( 6 ) default NULL ,
UNIQUE KEY 'id' ( 'id' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Загружаем таблицу primer_kladr в базу данных и приступаем к программированию примера. В файле index.php создадим первый уровень вложения — регионы и добавочно еще три пустых блока для подгрузки полей выбора нижестоящих уровней. Кнопка **Выбрать** неактивна до выбора нижнего уровня. По событию onchange происходит вызов хајаx-функции Select_Region с передачей всех значений формы. Каждое поле select имеет значение, равное значению КЛАДР для этого пункта. Для значений, предлагающих выбор (например "Выберите регион"), оно равно нулю. Перед вызовом функции предварительно устанавливается значение поля number, равное уровню элемента select, вызывающего функцию. Содержимое файла index.php приведено в листинге 2.9.

Листинг 2.9

```
<?php
    // подключение библиотеки
    require_once ("xajax_core/xajax.inc.php");
    // подключение внешних файлов
    require_once ("select_region.php");
    require_once ("result_select.php");
    // создать новый хаjax-объект
    $xajax = new xajax();
    // регистрация функций
    $xajax->register(XAJAX_FUNCTION,"Select_Region");
    $xajax->register(XAJAX_FUNCTION,"Result_Select");
    // разрешаем обрабатывать хаjax асинхронные запросы
    $xajax->processRequest();
    // подключение к базе данных
    require_once("mybaza.php");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
    charset=windows-1251">
<title> Пример 1 (глава 2) к книге "ХАЈАХ" </title>
<?php $xajax->printJavascript(''); ?>

</head>
<body>
<!-- шапка -->
<div id=header1><b>Примеры к книге "ХАЈАХ" (глава 2 пример 1)<br>
    Динамически подгружаемые select-элементы </b></div>
<br>
<!-- Форма -->
<form id='FormSelectRegion' method='post' action='javascript:void(null);'
    onsubmit='
        xajax.$("ButtonFormSelectRegion").disabled=true;
        xajax.$("ButtonFormSelectRegion").value="Подождите...";
        xajax_Result_Select(document.getElementById("vibor").value);
        enctype="multipart/form-data";>
<input type='hidden' id='number' name='number' value='0'><br>
<input type='hidden' id='vibor' name='vibor' value='0'><br>
<b> Выбор населенного пункта </b><br>
<div id='divselectregion1'>
```

```

<select name=selectregion1 id='selectregion1' onchange='
document.getElementById("number").value=1;
xajax_Select_Region(xajax.getFormValues("FormSelectRegion"));'>
<option value="0">Выберите регион

<?php
// получить первый уровень – регионы
$query1="SELECT id,name,socr FROM ".TABLE1." WHERE id_rayon=0 &&
        id_town=0 && id_punkt=0 ORDER BY id ASC";
$rez1=mysql_query($query1);
while($row1=mysql_fetch_row($rez1))
    $text1.= "<option value='".$row1[0]."' >".$row1[1]." ".$row1[2];
$text1.= "</select></div>";
$text1.= "<div id='divselectregion2'></div>";
$text1.= "<div id='divselectregion3'></div>";
$text1.= "<div id='divselectregion4'></div>";
echo $text1;
?>
<input type='submit' id='ButtonFormSelectRegion' value='Выбрать'
       disabled=true>
</form>
<!--Для вывода результата выбора -->
<div id='div_result'></div>
</body>
</html>

```

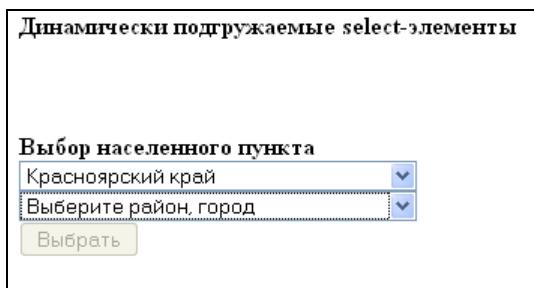


Рис. 2.11. Динамическая подгрузка в элемент select

Функция `Select_Region`, расположенная в файле `select_region.php`, получает значения формы **FormSelectRegion** и в зависимости от значения поля `number` выбирает запрос к базе данных. Из базы выбираются значения для следующего уровня `select`: для региона — районы и города регионального значения; для районов — населенные пункты, подчиненные району; для городов — объекты, подчиненные городу (если есть). По запросу формируется элемент `select` для нижестоящего уровня и динамически подгружается в блок (рис. 2.11). При этом поле `vibor`, равное текущему выбранному значению, устанавливается в нуль. Если объ-

ект КЛАДР не имеет нижестоящих объектов, активируется кнопка **Выбрать** и значение поля `vibor` устанавливается равным значению объекта КЛАДР.

Содержимое файла `select_region.php` приведено в листинге 2.10.

Листинг 2.10

```
<?php
function Select_Region($Id)
{
    $objResponse = new xajaxResponse();
    require_once("mybaza.php");

    switch($Id[number])
    {
        case 1:
            $objResponse->assign("vibor","value",$Id[selectregion1]);
            if($Id[selectregion1]>0)
            {
                $query1="SELECT name,socr FROM ".TABLE1." WHERE
                    id=".$Id[selectregion1]." ";
                $text1=mysql_result(mysql_query($query1),0,"name")." ";
                $text1.=mysql_result(mysql_query($query1),0,"socr");
                $text2=<select name=selectregion2 id='selectregion2'
                    onchange=
                        document.getElementById(\"number\").value=2;
                    xajax_Select_Region(xajax.getFormValues
                        (\\"FormSelectRegion\\"));'>;
                $text2.=<option value='0' selected>Выберите район, город";
                $objResponse->assign("vibor","value","0");
                $query2="SELECT id,name,socr,id_rayon FROM ".TABLE1."
                    WHERE id_punkt=0
                    && id_region=".substr($Id[selectregion1],1,2)." "
                    && (id_rayon=0 || id_town=0) && (id_rayon>0 || id_town>0)
                    ORDER BY id ASC";
                $rez2=mysql_query($query2);
                while($row2=mysql_fetch_row($rez2))
                {
                    if($row2[3]>0)
                        $text2.=<option value=". $row2[0]." >".$row2[1]."
                            ".$row2[2];
                    else
                        $text2.=<option value=". $row2[0]." >".$row2[2]."
                            ".$row2[1];
                }
            }
    }
}
```

```
$text2.="</select>";
$objResponse->assign("divselectregion2","innerHTML",$text2);
$objResponse->assign("divselectregion3","innerHTML",
"<table></table>");
$objResponse->assign("divselectregion4","innerHTML",
"<table></table>");
$objResponse->assign("ButtonFormSelectRegion","disabled",
true);
}
else //
{
    $objResponse->assign("vibor","value","0");
    $objResponse->assign
    ("divselectregion2","innerHTML","<table></table>");
    $objResponse->assign
    ("divselectregion3","innerHTML","<table></table>");
    $objResponse->assign
    ("divselectregion4","innerHTML","<table></table>");
    $objResponse->assign
    ("ButtonFormSelectRegion","disabled",true);
}
break;
case 2:
    $objResponse->assign("vibor","value",$Id[selectregion2]);
    if($Id[selectregion2]>0)
    {
        if(substr($Id[selectregion2],3,3)<>"000") // район
        {
            $text2=<select name=selectregion3 id='selectregion3'
onchange='
document.getElementById(\"number\").value=3;
xajax_Select_Region(xajax.getFormValues
(\"FormSelectRegion\"));'>;
            $objResponse->assign("vibor","value","0");
            $text2.= "<option value='0' selected>Выберите нас.пункт";
            $query2="SELECT id,name,socr FROM ".TABLE1." WHERE
                id_region=".substr($Id[selectregion2],1,2).""
                && id_rayon=".substr($Id[selectregion2],3,3)."
                && (id_punkt>0 || id_town>0)
                ORDER BY id ASC ";
            $rez2=mysql_query($query2);
            while($row2=mysql_fetch_row($rez2))
            {
                $text2.= "<option value='".$row2[0]."'>".$row2[2]."";
            }
        }
    }
}
```

```
    ".$row2[1];
}
$text2.="</select>";
$objResponse->assign
("divselectregion3","innerHTML",$text2);
$objResponse->assign
("divselectregion4","innerHTML",<table></table>");
$objResponse->assign
("ButtonFormSelectRegion","disabled",true);
}
else // город областного подчинения
{
$query2="SELECT id,name,socr FROM ".TABLE1." WHERE
        id_region=".substr($Id[selectregion2],1,2).""
        && id_rayon=0
        && id_town=".substr($Id[selectregion2],6,3)."
        ORDER BY id ASC";
$rez2=mysql_query($query2);
if(mysql_num_rows($rez2)==1) // нет вложенных
{
    $objResponse->assign
    ("ButtonFormSelectRegion","disabled",false);
    $objResponse->assign
    ("divselectregion3","innerHTML",<table></table>);
    $objResponse->assign
    ("divselectregion4","innerHTML",<table></table>);
}
else
{
    $text2=<select name=selectregion3 id='selectregion3'
onchange='
document.getElementById(\"number\").value=3;
xajax_Select_Region(xajax.getFormValues
(\\"FormSelectRegion\\"));'>;
    $objResponse->assign("vibor","value","0");
    $text2."<option value='0' selected>Выберите далее";
    while($row2=mysql_fetch_row($rez2))
    {
        $text2.= "<option value='".$row2[0]."'>".$row2[2].""
        ".$row2[1];
    }
    $text2.="</select>";
    $objResponse->assign
    ("ButtonFormSelectRegion","disabled",true);
```

```
    $objResponse->assign
    ("divselectregion3","innerHTML",$text2);
    $objResponse->assign
    ("divselectregion4","innerHTML","<table></table>");
}
}
else // {
{
    $objResponse->assign
    ("ButtonFormSelectRegion","disabled",true);
    $objResponse->assign("vibor","value","0");
    $objResponse->assign
    ("divselectregion3","innerHTML","<table></table>");
    $objResponse->assign
    ("divselectregion4","innerHTML","<table></table>");
}
break;
case 3:
if($Id[selectregion3]>0)
{
    $objResponse->assign
    ("ButtonFormSelectRegion","disabled",false);
    $objResponse->assign("vibor","value",$Id[selectregion3]);
    $objResponse->assign
    ("divselectregion4","innerHTML","<table></table>");
}
else
{
    $objResponse->assign("ButtonFormSelectRegion","disabled",true);
    $objResponse->assign("vibor","value","0");
    $objResponse->assign
    ("divselectregion4","innerHTML","<table></table>");
}
default: break;
}
return $objResponse;
}
?>
```

При нажатии на кнопку **Выбрать** происходит вызов хаях-функции `Result_Select`. Функции передается значение объекта КЛАДР из поля `vibor`. По этому коду формируем полный путь к объекту КЛАДР и выводим результат на страницу (рис. 2.12). Функция `Result_Select` расположена в файле `result_select.php`.

(листинг 2.11). Посмотреть в сети Интернет пример использования КЛАДР вы можете по адресу www.bazakatalogov.ru.

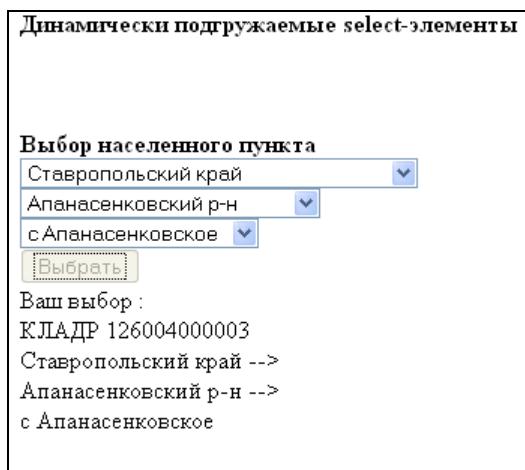


Рис. 2.12. Вывод результата выбора

Листинг 2.11

```
<?php
function Result_Select($Id)
{
    $objResponse = new xajaxResponse();
    require_once("mybaza.php");

    $text="Ваш выбор :<br>";
    $text.="КЛАДР ".$Id."<br>";
    $id_region=substr($Id,1,2);
    $id_rayon=substr($Id,3,3);
    $id_town=substr($Id,6,3);
    $id_punkt=substr($Id,9,3);
    // регион
    $query1="SELECT name,socr FROM ".TABLE1." WHERE
        id_region='".$id_region."'";
    $text.=mysql_result(mysql_query($query1),0,"name");
    $text.=".mysql_result(mysql_query($query1),0,"socr");
    // район
    if($id_rayon>0)
    {
        $query1="SELECT name,socr FROM ".TABLE1." WHERE
            id_region='".$id_region."' && id_rayon='".$id_rayon."'";
        $text.=mysql_result(mysql_query($query1),0,"name");
        $text.=".mysql_result(mysql_query($query1),0,"socr");
    }
}
```

```
$text.=" --> <br>".mysql_result(mysql_query($query1),0,"name");
$text.= " .mysql_result(mysql_query($query1),0,"socr");
}

// город
if($id_town>0)
{
$query1="SELECT name,socr FROM ".TABLE1." WHERE
    id_region='".$id_region."' && id_rayon='".$id_rayon."'
    && id_town='".$id_town."'";
$text.=" --> <br>".mysql_result(mysql_query($query1),0,"socr");
$text.= " .mysql_result(mysql_query($query1),0,"name");
}

// нас.пункт
if($id_punkt>0)
{
$query1="SELECT name,socr FROM ".TABLE1." WHERE
    id_region='".$id_region."' && id_rayon='".$id_rayon."'
    && id_town='".$id_town."' && id_punkt='".$id_punkt."'";
$text.=" --> <br>".mysql_result(mysql_query($query1),0,"socr");
$text.= " .mysql_result(mysql_query($query1),0,"name");
}

$objResponse->assign("div_result","innerHTML",$text);
$objResponse->assign("ButtonFormSelectRegion","value","Выбрать");
$objResponse->assign("ButtonFormSelectRegion","disabled",false);

return $objResponse;
}
?>
```

2.3.3. Многоуровневый неоднородный каталог

Представим, что на сайте большого магазина есть каталог, который имеет сложную многоуровневую структуру. Допустим, что количество уровней вложенности меняется от одной товарной группы к другой. Это напоминает путешествие по папкам в программе Проводник. Чтобы этот пример можно было сразу использовать в своих проектах, создадим его на основе базы данных MySQL. Файлы примера расположены на прилагаемом компакт-диске в папке book_primers/2-3.

Создадим в базе данных таблицу `primer_katalog`. Дамп для создания таблицы приведен в листинге 2.12.

Структура таблицы `primer_katalog`:

- `id` — первичный ключ;
- `id_parent` — ID родительской категории;

- name — наименование товара (категории);
- type — тип (категория, товар).

Листинг 2.12

```
CREATE TABLE 'book_xajax1'.'primer_katalog' (
'id' int( 5 ) NOT NULL AUTO_INCREMENT ,
'id_parent' int( 5 ) NOT NULL default '0',
'name' varchar( 30 ) NOT NULL default '',
'type' set( 'tovar', 'kategory' ) NOT NULL default 'kategory',
UNIQUE KEY 'id' ( 'id' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Загружаем таблицу primer_katalog в базу данных и приступаем к программированию примера. В файле index.php создадим первый уровень каталога. Для этого из базы данных выбираем позиции с id_parent=0. Для каждого элемента создаем блок с id=menu.\$id, где \$id — ID элемента в базе данных. Элементы у нас двух типов — каталог и товар, для каждого из них своя иконка. Кроме того для каталога может быть два вида иконок (для нераскрытый — open_dir.ico, для раскрытой — close_dir.ico). При нажатии на иконку каталога происходит вызов хаях-функции Open_Kategory либо Close_Kategory в зависимости от вида иконки. Содержимое файла index.php приведено в листинге 2.13.

Листинг 2.13

```
<?php
// подключение библиотеки
require_once ("xajax_core/xajax.inc.php");
// подключение внешних файлов
require_once ("open_kategory.php");
require_once ("close_kategory.php");
// создать новый хаях-объект
$xajax = new xajax();
// регистрация функций
$xajax->register(XAJAX_FUNCTION,"Open_Kategory");
$xajax->register(XAJAX_FUNCTION,"Close_Kategory");
// разрешаем обрабатывать хаях асинхронные запросы
$xajax->processRequest();
// подключение к базе данных
require_once ("mybaza.php");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title> Пример 3 (глава 2) к книге "ХАЖАХ" </title>

<?php $xajax->printJavascript(''); ?>

<style type="text/css">
    img { border-style: none }
    div.menu { margin-left:16px;font-size:small;
    vertical-align:center }
</style>
</head>
<body>
<!-- шапка -->
<div id=header1><b>Примеры к книге "ХАЖАХ" (глава 2 пример 3)<br>
    Многоуровневый неоднородный каталог </b></div>
<br>
<div id=menu0 class=menu>
<!-- Вывод первого уровня -->
<?php
    $text1="";
    $query1="SELECT * FROM ".TABLE1." WHERE id_parent=0
        ORDER BY id ASC";
    $rez1=mysql_query($query1);
    while($row1=mysql_fetch_assoc($rez1))
    {
        if($row1[type]=='kategory')
            $text1.= "<div class='menu' id='menu".$row1[id]."'>
                <a href='javascript:void();' onclick='
                    xajax_Open_Kategory(\".$row1[id].\")'>
                    <img src='open_dir.ico'></a>&ampnbsp&ampnbsp&ampnbsp
                    ".$row1[name]."</div>";
        else
            $text1.= "<div class='menu' id='menu".$row1[id]."'>
                <a href='javascript:void();' onclick=''
                    <img src='tovar.ico'></a>&ampnbsp&ampnbsp&ampnbsp
                    ".$row1[name]."</div>";
    }
    echo $text1;
    ?>
</div>
</body>
</html>
```

Функция `open_Kategory`, расположенная в файле `open_kategory.php` (листинг 2.14), вызывается для раскрытия содержимого элемента каталога. При этом

передается ID в базе данных раскрываемой категории. Из базы выбираем данные с id_parent, равным ID раскрываемой категории. Формируются блоки для каждого элемента и записываются в конец блока раскрываемого элемента.

Листинг 2.14

```
<?php
function Open_Kategory($Id)
{
    $objResponse = new xajaxResponse();

    require_once ("mybaza.php");
    $text1="";
    // текущая категория
    $query0="SELECT * FROM ".TABLE1." WHERE id='".$Id."'";
    $rez0=mysql_query($query0);
    $row0=mysql_fetch_assoc($rez0);
    $text1.="

```

Функция Close_Kategory, расположенная в файле close_category.php (листинг 2.15), вызывается для закрытия содержимого каталога. В блок для закры-

ваемого элемента записываем только его контент. Данные для вложенных элементов при этом исчезают. Вид каталога иллюстрирует рис. 2.13.

ЗАМЕЧАНИЕ

На рис. 2.13 видно постоянное смещение вправо на 16 px для позиций нижестоящего уровня относительно вышестоящего. Это достигается установкой стиля для блоков `div.menu {margin-left:16px}`.

Листинг 2.15

```
<?php
function Close_Kategory($Id)
{
    $objResponse = new xajaxResponse();

    require_once("mybaza.php");
    $text1="";
    // удалить вложенные категории
    $query1="SELECT * FROM ".TABLE1." WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    $text1.="
```

Многоуровневый неоднородный каталог



Рис. 2.13. Многоуровневый каталог

2.3.4. Динамическое управление числом полей формы

На сайтах часто приходится решать задачу добавления (или удаления) полей для уже существующей формы. Создадим форму заказа экскурсии для группы людей. В форму необходимо вводить ФИО каждого человека. Поля для ввода ФИО будем добавлять динамически при нажатии ссылки **Добавить человека**. Также может возникнуть необходимость удалить человека из списка. Файлы примера расположены на компакт-диске в папке book_primers/2-4. В файле index.php создадим форму с необходимыми полями. Начальный вид страницы представлен на рис. 2.14. Содержимое файла index.php приведено в листинге 2.16. Экскурсии выбираем из выпадающего списка. Рассматриваемый пример будем делать без использования баз данных, поэтому каждое значение `option` будет содержать цену и название экскурсии, разделенные символом `:`. Первое поле `input` для ввода ФИО имеет `id=name1`. Последующие созданные поля будут иметь `id=name2, name3` и т. д. Так как эти поля могут создаваться и удаляться в произвольном порядке, необходимо иметь указатель на последний созданный ID. Оно будет храниться в скрытом поле `input type="hidden" id="counted"`. Необходимо иметь скрытое поле с указанием удаляемого элемента, его значение будет устанавливаться при нажатии на ссылку **Удалить** перед передачей значений формы хаях-функции `Delete_Input`.

Примеры к книге "ХАКАРД" (глава 2 пример 4)
Динамическое управление количеством полей формы

Экскурсии на 2010-09-03

Стоимость экскурсии (руб.)
650
Число
1
Сумма (руб.)
650
Телефон для контакта

Состав группы (минимум 1 человек):

[Добавить человека](#)

Рис. 2.14. Первоначальный вид формы

Листинг 2.16

```

<?php
    // подключение библиотеки
    require_once ("xajax_core/xajax.inc.php");
    // подключение внешних файлов
    require_once ("add_input.php");
    require_once ("delete_input.php");
    require_once ("new_path.php");
    require_once ("result.php");
    // создать новый хаях-объект
    $xajax = new xajax();
    // регистрация функций
    $xajax->register(XAJAX_FUNCTION,"Add_Input");
    $xajax->register(XAJAX_FUNCTION,"Delete_Input");
    $xajax->register(XAJAX_FUNCTION,"New_Path");
    $xajax->register(XAJAX_FUNCTION,"Result");
    // разрешаем обрабатывать хаях асинхронные запросы
    $xajax->processRequest();
    // подключение к базе данных
    require_once("mybaza.php");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
        <title> Пример 4 (глава 2) к книге "ХАЯХ" </title>
        <?php $xajax->printJavascript(''); ?>
    </head>
    <body>
        <!-- Шапка -->
        <div id=header1><b>Примеры к книге "ХАЯХ" (глава 2 пример 4)<br>
            Динамическое управление количеством полей формы </b></div>
        <br>
        <!-- Форма -->
        <div id='div1'>
            <form id='Form1' action='javascript:void(null);' onsubmit='
                xajax.$("Button_Form1").disabled=true;
                xajax.$("Button_Form1").value="Подождите...";'
                xajax_Result(xajax.getFormValues("Form1"));'>
                <table>
                    <!-- маршрут -->
                    Экскурсии на
                <?php

```

```
echo date('Y-m-d', strtotime("today + 1 day"));
?>
<br>
<select name='select_path' id='select_path'
    onchange='xajax_New_Path(xajax.getFormValues("Form1"));'>
<option value="550;Домбай" selected>Домбай 650 руб.
<option value="700;Архыз">Архыз 700 руб.
<option value="400;Медовые водопады">Медовые водопады 400 руб.
<option value="650;Приэльбрусье">Приэльбрусье 650 руб.
<option value="450;Чегемские водопады">Чегемские водопады 450 руб.
</select>
<br><br>
<!-- скрытый счетчик -->
<input type='hidden' name='count' id='count' value='1'>
<input type='hidden' name='pay' id='pay' value='650'>
<input type='hidden' name='countid' id='countid' value='1'>
<input type='hidden' name='delete' id='delete' value='0'>
<!-- вывод информации -->
Стоимость экскурсии (руб.)
<div name='pay1' id='pay1'>650</div>
Кол-во
<div name='people' id='people'>1</div>
Сумма (руб.)
<div name='payall' id='payall'>650</div>
Телефон для контакта<br>
<input type='text' name='phone' id='phone' size=12 maxlength=12><br><br>
<!-- состав группы -->
Состав группы (минимум 1 человек): <br>
<br>
<input type='text' name='name1' id='name1' value='Введите ФИО' size=50 maxlength=50><br><br>
<!-- ссылка Добавить -->
<a href='javascript:void();' onclick='
    xajax_Add_Input(xajax.getFormValues("Form1"));'>
Добавить человека</a>
<br><br>
<input type='submit' id='Button_Form1' value='Подать заявку ->' >
</div>
<div id='result'>
</div>
</body>
</html>
```

Примеры к книге "ХАJAX" (глава 2 пример 4)
Динамическое управление количеством полей формы

Экскурсии на 2010-09-03
Домбай 650 руб.

Стоимость экскурсии (руб.)
650
Число
3
Сумма (руб.)
1950
Телефон для контакта
9187855718

Состав группы (минимум 1 человек):

Иванов Иван	Удалить
Петрова Анна	Удалить
Сергеев Николай	Удалить

[Добавить человека](#)

[Подать заявку →](#)

Рис. 2.15. Вид формы после добавления поля input

При нажатии на ссылку **Добавить человека**, вызывается хаях-функция `Add_Input`, расположенная в файле `add_input.php` (листинг 2.17). При этом для добавления поля `input` используем `response`-метод `insertInputAfter`. Затем добавляем элементы `
` и ``, в элемент `` вставляем ссылку на удаление элемента `input` и пересчитываем заново сумму платежа. Вид формы после добавления нового поля иллюстрирует рис. 2.15.

ЗАМЕЧАНИЕ

Элементы `` и даже `
` должны иметь свой уникальный ID, чтобы при удалении поля не изменился вид формы.

Листинг 2.17

```
<?php
function Add_Input($Id)
{
    $objResponse = new xajaxResponse();
    // создать input
    $countid=$Id[$countid]+1;
    $str="<input type='text' value='><br><span id='del$Id[$countid]'>Удалить</span>";
    $objResponse->insertInputAfter($str,$Id[$countid]);
}
```

```

$objResponse->insertInputAfter
  ("name1","input","name".$countid,"name".$countid);
// вставить <br>
$objResponse->insertAfter("name1","br","br".$countid);
// установить стили
$objResponse->assign("name".$countid,"size","50");
$objResponse->assign("name".$countid,"value","Введите ФИО");
// вставить <span> для ссылки удалить
$objResponse->insertAfter("name".$countid,"span","span".$countid);
// ссылка удалить
$a=<a href='javascript:void()';
  onclick='document.getElementById(\"delete\").value='".$countid."';
  xajax_Delete_Input(xajax.getFormValues (\\"Form1\\"))'>Удалить</a>";
$objResponse->assign("span".$countid,"innerHTML",$a);
// установить новый countid
$objResponse->assign("countid","value",$countid);
// установить новый count
$objResponse->assign("count","value",$Id[count]+1);
// получить цену
// $pay1=substr($Id[select_path],0,strpos($Id[select_path],";"));
// установить цену, кол-во, сумму
$objResponse->assign("pay1","innerHTML",$Id[pay]);
$objResponse->assign("people","innerHTML",$Id[count]+1);
$objResponse->assign("payall","innerHTML",$Id[pay]*($Id[count]+1));

return $objResponse;
}
?>
```

При нажатии на ссылку **Удалить**, вызывается хайх-функция `Delete_Input`, расположенная в файле `delete_input.php` (листинг 2.18). При этом для удаления поля `input` используем `response`-метод `remove`. Необходимо удалить не только элемент `input`, но и элементы `
` и ``, а также заново пересчитать сумму платежа.

ЗАМЕЧАНИЕ

При нажатии на ссылку удаления перед вызовом функции `Delete_Input`, необходимо изменить значение скрытого поля `delete` на ID удаляемого элемента `input`.

Листинг 2.18

```

<?php
function Delete_Input($Id)
{
```

```
$objResponse = new xajaxResponse();
// удалить input
$objResponse->remove("name".$Id[delete]);
// удалить ссылку
$objResponse->remove("span".$Id[delete]);
// удалить br
$objResponse->remove("br".$Id[delete]);
// установить новый count
$objResponse->assign("count","value",$Id[count]-1);
// получить цену
$pay1=substr($Id[select_path],0,strpos($Id[select_path],";"));
// установить цену, кол-во, сумму
$objResponse->assign("pay1","innerHTML",$pay1);
$objResponse->assign("people","innerHTML",$Id[count]-1);
$objResponse->assign("payall","innerHTML",$pay1*($Id[count]-1));

return $objResponse;
}
?>
```

При изменении экскурсии по событию `onchange` вызывается хайп-функция `New_Path`, расположенная в файле `new_path.php` (листинг 2.19). При этом необходимо пересчитать сумму платежа. Вид формы после выполнения хайп-функции иллюстрирует рис. 2.16.

Листинг 2.19

```
<?php
function New_Path($Id)
{
$objResponse = new xajaxResponse();
// получить цену
$pay1=substr($Id[select_path],0,strpos($Id[select_path],";"));
// установить новый pay
$objResponse->assign("pay","value",$pay1);
// установить цену, кол-во, сумму
$objResponse->assign("pay1","innerHTML",$pay1);
$objResponse->assign("people","innerHTML",$Id[count]);
$objResponse->assign("payall","innerHTML",$pay1*$Id[count]);

return $objResponse;
}
?>
```

Примеры к книге "ХАЈАХ" (глава 2 пример 4)
Динамическое управление количеством полей формы

Экскурсии на 2010-09-03

Стоимость экскурсии (руб.)
 450

Число
 3

Сумма (руб.)
 1350

Телефон для контакта

Состав группы (минимум 1 человек):

Иванов Иван	Удалить
Петрова Анна	Удалить
Сергеев Николай	Удалить

[Добавить человека](#)

Рис. 2.16. Изменение данных после выбора другой экскурсии

Иванов Иван	Удалить
Михеева Инна	Удалить
Петрова Анна	Удалить

[Добавить человека](#)

Ваша заявка принята
 Экскурсия - Чегемские водопады
 Стоимость 450 руб.
 В группе 3 человек
 Список:
 1. Иванов Иван
 2. Петрова Анна
 3. Михеева Инна
 Общая сумма 1350 руб.
 С Вами свяжется менеджер по указанному Вами телефону - 9187855718

Рис. 2.17. Вывод результата заявки

При нажатии на кнопку **Подать заявку** вызываем хайх-функцию `Result`, расположенную в файле `result.php`, передавая при этом ей все значения формы. В реальном проекте эту информацию следует заносить в базу данных, в примере мы формируем контент подтверждения о приеме заказа в блоке `result` (рис. 2.17). Содержимое файла `result.php` приведено в листинге 2.20.

ЗАМЕЧАНИЕ

Для переноса блока вывода результата в зону видимости используем JavaScript-функцию `scrollIntoView()`. При этом используется `response`-метод `script`.

Листинг 2.20

```
<?php
function Result($Id)
{
    $objResponse = new xajaxResponse();
    $text="";
    $text="Ваша заявка принята<br>";
    // получить название экскурсии и цену
    $pay=substr($Id['select_path'],0,strpos($Id['select_path'],";"));
    $path=str_replace($pay.",","", $Id['select_path']);
    $text.="Экскурсия - ".$path."<br>";
    $text.="Стоимость ".$Id['pay']." руб. <br>";
    // список людей
    $text.="В группе ".$Id['count']."' человек <br>";
    $text.="Список : <br>";
    for($i=1,$j=1;$i<=$Id['countid'];$i++)
    {
        if(isset($Id['name'].$i))
        {
            $text.=$j.". ".$Id['name'].$i."<br>";
            $j++;
        }
    }
    // общая сумма
    $text.= "<br>Общая сумма ".$Id['pay']*$Id['count']."' руб. <br>";
    // телефон
    $text.= "<br>С Вами свяжется менеджер по указанному
    Вами телефону - ".$Id['phone'];
    // вывести в блок result
    $objResponse->assign("result","innerHTML",$text);
    // кнопку установить
```

```
$objResponse->assign("Button_Form1","value","Подать заявку ->");  
$objResponse->assign("Button_Form1","disabled",false);  
// блок в поле видимости  
$objResponse->script("document.getElementById  
('result').scrollIntoView();");  
  
return $objResponse;  
}  
?  
>
```

2.4. Фреймворк jQuery

Что такое jQuery? jQuery — это JavaScript-библиотека, которая появилась в январе 2006 года. На сайте разработчиков лозунг: "jQuery is designed to change the way that you write JavaScript". Если переводить это буквально, то получится примерно следующее: "jQuery разработан, чтобы изменить путь, которым вы пишете на JavaScript". jQuery помогает легко получать доступ к любому элементу (набору элементов) объектной модели документа (DOM), обращаться к атрибутам и содержимому элементов DOM и конечно манипулировать ими. Причем благодаря своему интуитивно понятному синтаксису, схожему в чем-то с CSS1, CSS2 и XPath, эта работа становится не просто легкой, а я бы сказал, приятной. Также библиотека jQuery предоставляет удобный API (интерфейс программирования приложений) по работе с AJAX. Саму библиотеку можно скачать на сайте разработчиков <http://jquery.com>. На момент написания книги доступна версия 1.4.2. На сайте представлена хорошо проработанная документация, масса примеров, подробнейшее описание и большое количество плагинов (основные включены отдельными файлами в архив библиотеки), предназначенных для создания на их основе элементов пользовательских интерфейсов.

2.4.1. Возможности jQuery

Фреймворк имеет следующие возможности:

- переход по дереву DOM;
- события;
- визуальные эффекты;
- AJAX-дополнения;
- JavaScript-плагины.

Библиотека jQuery содержит функционал, полезный для максимально широкого круга задач. Тем не менее, разработчики библиотеки не ставили задачу совмещения в jQuery функций, которые подошли бы всюду, поскольку тогда объем кода неоправданно возрастает. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно тот JavaScript-функционал, который на нем был бы востребован.

2.4.2. Использование jQuery

Для скачивания библиотеки jQuery заходим по адресу http://docs.jquery.com/Downloading_jQuery и нажимаем правой кнопкой мыши на ссылку **Uncompressed**. Выбираем пункт **Сохранить Объект как** (рис. 2.18) и сохраняем файл на компьютере.

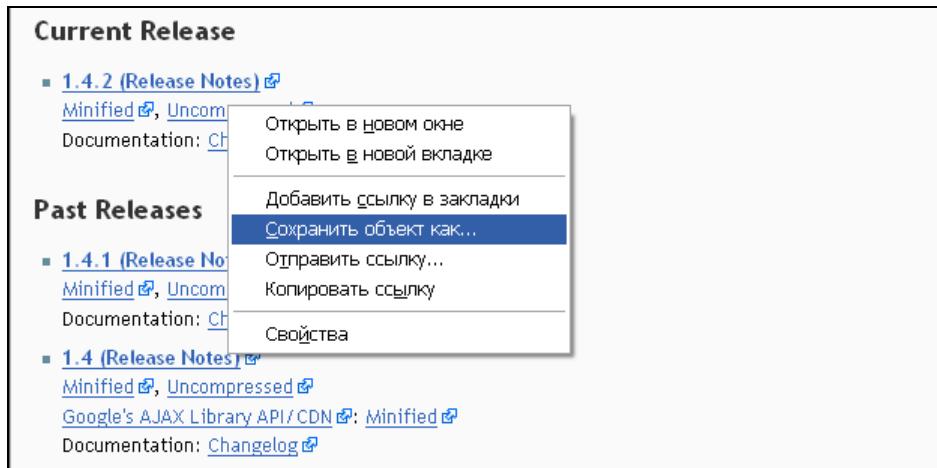


Рис. 2.18. Скачиваем файл библиотеки jQuery

jQuery, как правило, включается в Web-страницу как один внешний JavaScript-файл:

```
<head>
<script type="text/javascript" src="js/jquery-1.4.2.js"></script>
</head>
```

Работу с jQuery можно разделить на два этапа:

- получение jQuery-объекта с помощью функции `$()`;
- вызов глобальных методов у объекта `$`.

2.4.2.1. Функция \$

Взаимодействие с jQuery осуществляется с помощью функции `$`. Если на сайте применяются другие JavaScript-библиотеки, где `$` может использоваться для своих нужд, то ее можно заменить синонимом — `jQuery`. Такой способ считается более правильным, а чтобы код не получался слишком громоздким, можно записать его следующим образом:

```
jQuery(function($)
{
    // Тут код скрипта, где в $ будет jQuery
})
```

Вне зависимости от параметров, переданных в функцию, она вернет список объектов, над которым уже определены все доступные jQuery-функции (а их немало). Это позволяет работать с любыми объектами (уже имеющимися на странице, созданными динамически или полученными через AJAX) так, будто это одни и те же элементы, уже существующие на странице. В jQuery реализован очень интересный механизм поиска элементов на основе CSS и Xpath. Для нахождения требуемого элемента вы можете воспользоваться как механизмом селекторов CSS, так и запросами по документу в стиле Xpath.

2.4.2.2. Селекторы

Для того чтобы понимать, как работает селектор, все же необходимы базовые знания CSS, т. к. именно от принципов CSS отталкивается селектор jQuery. Список поддерживаемых селекторов CSS приведен в табл. 2.1.

Таблица 2.1. Поддерживаемые селекторы CSS

Селектор	Описание
*	Элемент E, являющийся n-м дочерним элементом своего родительского элемента
E	Элемент типа E
E:nth-child(n)	E, являющийся n-м дочерним элементом своего родительского элемента
E:first-child	E, являющийся первым дочерним элементом своего родительского элемента
E:last-child	E, являющийся последним дочерним элементом своего родительского элемента
E:only-child	E, являющийся единственным дочерним элементом своего родительского элемента
E:empty	E, у которого нет дочерних элементов (включая текстовые узлы)
E:enabled	Активный элемент E пользовательского интерфейса
E:disabled	Неактивный элемент E пользовательского интерфейса
E:checked	Отмеченный элемент E пользовательского интерфейса (например, радиокнопка)
E.warning	Элемент E с классом warning (class="warning")
E#myid	E с ID, равным "myid" (выберет максимум один элемент)
E:not(s)	E, не соответствующий простому селектору s
E F	F, являющийся потомком элемента E
E > F	F, являющимся дочерним элементом E
E + F	F, которому непосредственно предшествует E
E ~ F	F, которому предшествует E
E, F, G	Выбрать все элементы E, F и G

Примеры

```

$( '#element1' );
// выбор элемента с id=element1
$('.class1');

// выбор элементов с class=class1
$('div#element1');

// выбор элемента div с id=element1
$('div.class1');

// выбор элементов div с class=class1
$('div span');

// выбор всех span элементов в элементах div
$('div > a');

// выбор всех a элементов в элементах div, где span является прямым
// потомком div
$('div, a');

// выбор всех div и a элементов
$('a + img');

// выбор всех img элементов, перед которыми идут a элементы
$('a ~ img');

// выбор всех img элементов после первого элемента a

```

Селекторы атрибутов приведены в табл. 2.2. Их нужно записывать в стиле XPath, т. е. с предваряющим символом @.

Таблица 2.2. Селекторы атрибутов jQuery

Селектор	Описание
E[@foo]	Элемент E с атрибутом "foo"
E[@foo=bar]	E, у которого значение атрибута "foo" равно "bar"
E[@foo^=bar]	E, у которого значение атрибута "foo" начинается с "bar"
E[@foo\$=bar]	E, у которого значение атрибута "foo" оканчивается на "bar"
E[@foo*=bar]	E, у которого значение атрибута "foo" содержит "bar"

jQuery поддерживает некоторые нестандартные селекторы (табл. 2.3), использование которых облегчает жизнь.

Таблица 2.3. Собственные селекторы jQuery

Селектор	Описание
:even	Выбирает все четные элементы из коллекции
:odd	Все нечетные элементы из коллекции
:eq(N) и :nth(N)	Выбирает элемент с индексом N из коллекции

Таблица 2.3 (окончание)

Селектор	Описание
:gt (N)	Элементы с индексом большим, чем N
:lt (N)	Элементы с индексом меньшим, чем N
:first	Первый элемент из коллекции
:last	Последний элемент из коллекции
:parent	Все элементы, у которых есть дочерние элементы (включая текст)
:contains ('test')	Все элементы, которые содержат текст
:visible	Все видимые элементы (включая элементы со стилями "display", равными "block" или "inline", "visibility", равным "visible", а также элементы форм, не относящихся к типу "hidden")
:hidden	Выбирает все невидимые элементы (включая элементы со стилями "display", равными "none")

Примеры

```
$('div:even');
// выбираем четные div
$('div:odd');

// выбираем нечетные div
$('div:eq(N)');
// выбираем div, идущий под номером N в DOM
$('div:gt(N)');
// выбираем div, индекс которых больше, чем N в DOM
$('div:lt(N)');
// выбираем div, индекс которых меньше, чем N в DOM
$('div:contains(text)');
// выбираем div, содержащие текст
$('div:empty');
```

Для работы с элементами форм предусмотрен ряд селекторов, позволяющих осуществлять выбор с учетом типа элемента и фильтров (enabled/disabled/selected/checked) (табл. 2.4).

Таблица 2.4. Селекторы форм

Селектор	Описание
:input	Выбирает все элементы формы (input, select, textarea, button)
:text	Текстовые поля (type="text")
:password	Поля для ввода паролей (type="password")

Таблица 2.4 (окончание)

Селектор	Описание
:radio	Радиокнопки (<code>type="radio"</code>)
:checkbox	Флаговые поля (<code>type="checkbox"</code>)
:submit	Кнопки для отсылки формы (<code>type="submit"</code>)
:image	Изображения на форме (<code>type="image"</code>)
:reset	Кнопки очистки формы (<code>type="reset"</code>)
:button	Выбирает все остальные кнопки (<code>type="button"</code>)

Примеры

```
$(":text");
// выбор всех input-элементов с типом =text
$(":radio");
// выбор всех input-элементов с типом =radio
$("input:enabled");
// выбор всех включенных элементов input
$("input:checked");
// выбор всех отмеченных флажков
```

2.4.2.3. Методы jQuery

jQuery предлагает разработчику большое количество методов для манипуляции элементами документа и их свойствами:

- `append(content)` — добавить переданный элемент или выражение в конец выбранного элемента;
- `prepend(content)` — добавить переданный элемент или выражение в начало выбранного элемента;
- `appendTo(expr)` — добавить выбранный элемент в конец переданного элемента;
- `prependTo(expr)` — добавить выбранный элемент в начало переданного элемента;
- `attr(name)` — получить значение атрибута;
- `attr(params)` — установить значение атрибутов; атрибуты передаются в виде {`ключ1:значение1[, ключ2:значение2[, ...]]`};
- `attr(name, value)` — установить значение одного атрибута;
- `css(name)` — получить/установить значение отдельных параметров CSS;
- `css(params)` — установить значение отдельных параметров CSS;
- `css(name, value)` — установить значение одного параметра CSS;
- `text()` — получить текст элемента;
- `text(val)` — задать текст элемента;

- `html()` — получить HTML-код элемента;
- `html(val)` — задать HTML-код элемента;
- `empty()` — удалить все подэлементы текущего элемента.

Примеры

```
$("<b>Текст</b>").appendTo($("#div1"));
// добавить <b>Текст</b> в конец элемента с id=div1
$("#div1").empty();
// очистить содержимое элемента с id=div1
$("#div1").prepend("<b>Текст</b>");
// добавить <b>Текст</b> в начало элемента с id=div1
$("#div1").css({backgroundColor: "#F00",color: "#00F"});
// для элемента с id=div1 установить значение цвета и фона
```

Главная особенность большинства методов jQuery — возможность связывать их в цепочки. Методы, манипулирующие элементами документа, обычно возвращают эти объекты для дальнейшего использования.

Пример

```
// найти <select id="select1">...</select>
var sel = $("#select1");
// добавляем к нему <option value="1">Пример опции</option>
$("<option></option>")
// создаем требуемый элемент
.attr("value", 1)
// устанавливаем значение одного из его атрибутов
.html("Пример опции")
// записываем в него текст
.appendTo(sel);
// прикрепляем к уже существующему элементу
```

Таким образом, можно легко описать все действия, происходящие с выбранным элементом, не затрудняясь введением большого количества временных переменных.

2.4.2.4. Обработка событий в jQuery

Список событий, поддерживаемых в jQuery, приведен в табл. 2.5.

Таблица 2.5. Список событий jQuery

Событие	Описание
<code>blur()</code>	Вызывает событие <code>blur</code> для каждого элемента набора
<code>blur(функция)</code>	Назначает функцию к событию <code>blur</code> для каждого элемента набора

Таблица 2.5 (продолжение)

Событие	Описание
change ()	Вызывает событие change для каждого элемента набора
change (функция)	Назначает функцию к событию change для каждого элемента набора
click ()	Вызывает событие click для каждого элемента набора
click (функция)	Назначает функцию к событию click для каждого элемента набора
dblclick ()	Вызывает событие dblclick для каждого элемента набора
dblclick (функция)	Назначает функцию к событию dblclick для каждого элемента набора
error ()	Вызывает событие error для каждого элемента набора
error (функция)	Назначает функцию к событию error для каждого элемента набора
focus ()	Вызывает событие focus для каждого элемента набора
focus (функция)	Назначает функцию к событию focus для каждого элемента набора
keydown ()	Вызывает событие keydown для каждого элемента набора
keydown (функция)	Назначает функцию к событию keydown для каждого элемента набора
keypress ()	Вызывает событие keypress для каждого элемента набора
keypress (функция)	Назначает функцию к событию keypress для каждого элемента набора
keyup ()	Вызывает событие keyup для каждого элемента набора
keyup (функция)	Назначает функцию к событию keyup для каждого элемента набора
load (функция)	Назначает функцию к событию load для каждого элемента набора
mousedown (функция)	Назначает функцию к событию mousedown для каждого элемента набора
mouseenter (функция)	Назначает функцию к событию mouseenter для каждого элемента набора
mouseleave (функция)	Назначает функцию к событию mouseleave для каждого элемента набора
mousemove (функция)	Назначает функцию к событию mousemove для каждого элемента набора
mouseout (функция)	Назначает функцию к событию mouseout для каждого элемента набора

Таблица 2.5 (окончание)

Событие	Описание
mouseover (функция)	Назначает функцию к событию <code>mouseover</code> для каждого элемента набора
mouseout (функция)	Назначает функцию к событию <code>mouseout</code> для каждого элемента набора
resize (функция)	Назначает функцию к событию <code>resize</code> для каждого элемента набора
scroll (функция)	Назначает функцию к событию <code>scroll</code> для каждого элемента набора
<code>select()</code>	Вызывает событие <code>select</code> для каждого элемента набора
<code>select(функция)</code>	Назначает функцию к событию <code>select</code> для каждого элемента набора
<code>submit()</code>	Вызывает событие <code>submit</code> для каждого элемента набора
<code>submit(функция)</code>	Назначает функцию к событию <code>submit</code> для каждого элемента набора
<code>unload(функция)</code>	Назначает функцию к событию <code>unload</code> для каждого элемента набора

2.4.2.5. Эффекты в jQuery

Грамотная манипуляция свойствами элементов на странице позволила создателям JavaScript-библиотек реализовать визуальные эффекты, которые раньше были возможны только при использовании технологии Flash. Это плавное появление и скрытие объектов, плавное изменение различных свойств объектов (фонового цвета, размеров), реализация всевозможных элементов интерфейса (сплиттеров, деревьев, перетягиваемых объектов и сортируемых списков).

Методы jQuery для показа и скрытия элементов:

- `show([speed[, callback]])` — показать элемент;
- `hide([speed[, callback]])` — скрыть элемент;
- `fadeIn(speed[, callback])` — показать элемент путем изменения его прозрачности;
- `fadeOut(speed[, callback])` — скрыть элемент путем изменения его прозрачности;
- `slideDown(speed, callback)` — показать элемент, спустив его сверху;
- `slideUp(speed, callback)` — показать элемент, подняв его снизу.

Здесь `speed` — скорость в миллисекундах или одно из значений `"slow"` (600 мс) или `"fast"` (200 мс); `callback` — функция, которая будет вызвана после выполнения анимации.

2.4.3. PHP и jQuery

На сегодняшний день к библиотеке jQuery написано очень много интересных плагинов. Но написать большой проект на JavaScript очень сложно. Поддержка AJAX средствами JQuery не может полностью решить эту проблему, т. к. требует написания JavaScript-функций обработки ответов с сервера. Нам нужен PHP-фреймворк, который в требуемый момент будет осуществлять динамическую подгрузку библиотеки jQuery и плагинов на страницу. Для этой цели как раз и подойдет хаах. Рассмотрим примеры использования связки хаах и jQuery.

2.4.3.1. Динамическая подгрузка jQuery и плагина Carousel

Для начала создадим простой пример динамической подгрузки на страницу плагина jQuery. Плагин будем подгружать на страницу при нажатии на ссылку по AJAX, используя фреймворк хаах. Файлы примера находятся на компакт-диске в папке book_primers/2-5. Для загрузки примера наберите в браузере http://book_primers/2-5 (Денвер должен быть запущен). Вы увидите страницу, изображенную на рис. 2.19.

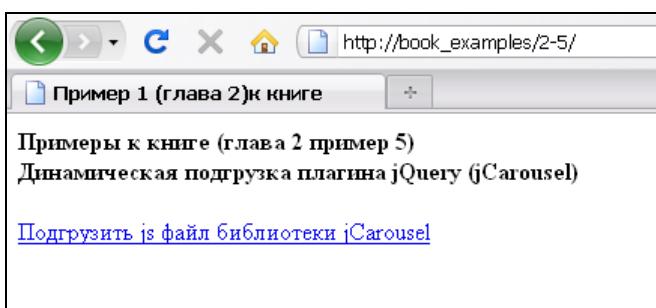


Рис. 2.19. Вид страницы при загрузке примера 2-5

При нажатии ссылки **Подгрузить js файл библиотеки jCarousel** подгружаем внешний файл библиотеки jCarousel (вызов хаах-функции с аргументом 1). Страница принимает вид, показанный на рис. 2.20.

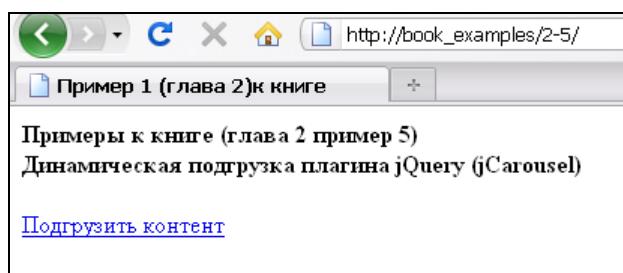


Рис. 2.20. Вид страницы после загрузки js библиотеки jCarousel



Рис. 2.21. Вид страницы после загрузки контента для плагина jCarousel

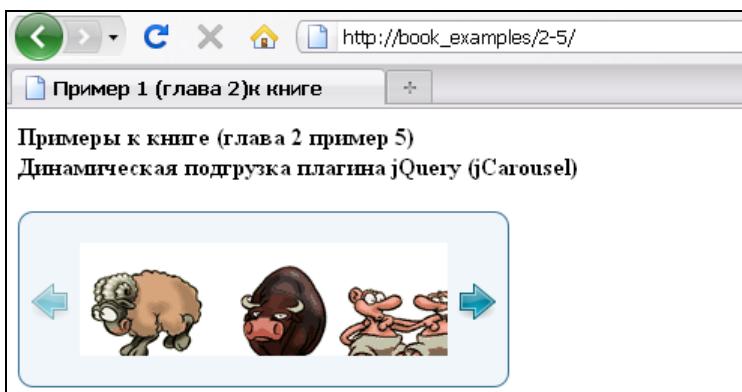


Рис. 2.22. Плагин jCarousel загружен

При нажатии ссылки **Подгрузить контент** подгружаем контент для плагина (вызов хайп-функции с аргументом 2). Вид страницы показан на рис. 2.21.

При нажатии ссылки **Запустить плагин** запускаем плагин (вызов хайп-функции с аргументом 2). Вид страницы показан на рис. 2.22.

Содержимое файла index.php приведено в листинге 2.21. Из файла идет запуск хайп-функции Plugin, которая расположена в файле plugin.php (листинг 2.22).

Листинг 2.21

```
<?php
require_once ("xajax_core/xajax.inc.php");
require_once ("plugin.php");
```

```
$xajax = new xajax();
// регистрация функций
$xajax->register(XAJAX_FUNCTION,"Plugin");
$xajax->processRequest();

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<link rel="stylesheet" type="text/css" href="js/skins/tango/skin.css" />
<title> Пример 1 (глава 2) к книге </title>
<script type="text/javascript" src="js/jquery-1.4.2.js"></script>
<?php $xajax->printJavascript(''); ?>
</head>
<body>
<!-- шапка -->
<div id=header1><b>Примеры к книге (глава 2 пример 5)<br>
    Динамическая подгрузка плагина jQuery (jCarousel)</b></div>
<br>
<!-- Форма -->
<div id='div0'>
<a href='javascript:void();' onclick='xajax_Plugin(1);'>
Подгрузить js-файл библиотеки jCarousel</a>
</div>
<div id='div1'><!-- Место для загрузки плагина-->
</div>
</body>
</html>
```

Листинг 2.22

```
<?php
// Подгрузка плагина jQuery
function Plugin($Id)
{
    $objResponse = new xajaxResponse();

    switch ($Id)
    {
        case 1:
            $objResponse->includeScript("js/jquery.jcarousel.js");
            $text=<a href='javascript:void();'
                onclick='xajax_Plugin(2);'
                Подгрузить контент</a>";
            $objResponse->printText($text);
    }
}
```

```

$objResponse->assign("div0","innerHTML",$text);
break;
case 2:
$text1="";
<ul id='mycarousel' class='jcarousel-skin-tango'>
<li><img src='img/img1.gif' /></li>
<li><img src='img/img2.gif' /></li>
<li><img src='img/img3.gif' /></li>
<li><img src='img/img4.gif' /></li>
<li><img src='img/img5.gif' /></li>
<li><img src='img/img6.gif' /></li>
<li><img src='img/img7.gif' /></li>
<li><img src='img/img8.gif' /></li>
<li><img src='img/img9.gif' /></li>
<li><img src='img/img10.gif' /></li>
<li><img src='img/img11.gif' /></li>
<li><img src='img/img12.gif' /></li>
</ul>;
$objResponse->assign("div1","innerHTML",$text1);
$text2=<a href='javascript:void();' onclick='xajax_Plugin(3);'>
Запустить плагин</a>;
$objResponse->assign("div0","innerHTML",$text2);
break;
case 3:
$script="jQuery('#mycarousel').jcarousel();";
$objResponse->script($script);
$text2="";
$objResponse->assign("div0","innerHTML",$text2);
break;
default: break;
}
return $objResponse;
}
?>

```

2.4.3.2. Совместное использование jQuery UI, виджетов Tabs и Accordion

Query UI — надстройка над JavaScript-библиотекой jQuery, помогающая создавать по-настоящему интерактивные Web-приложения. Рассмотрим виджеты Accordion и Tabs. На рис. 2.23 изображен пример с виджетом Accordion. Щелчок по заголовку скрывает/отображает содержимое, разбитое на логические секции. При отображении содержимого одной секции, открытая ранее секция обязательно закрывается.

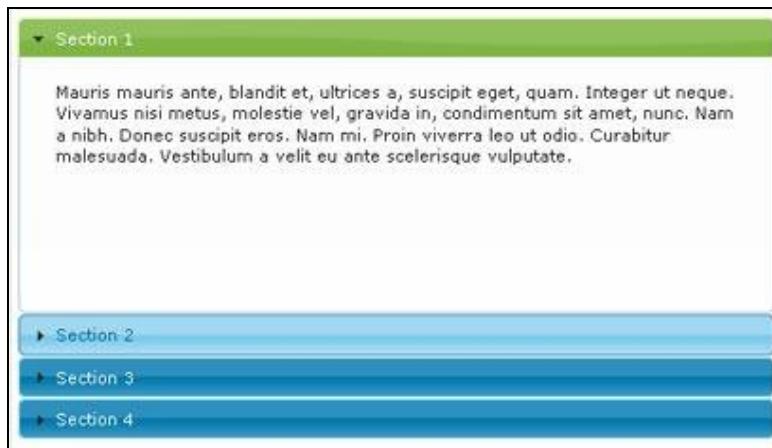


Рис. 2.23. jQueryUI, виджет Accordion

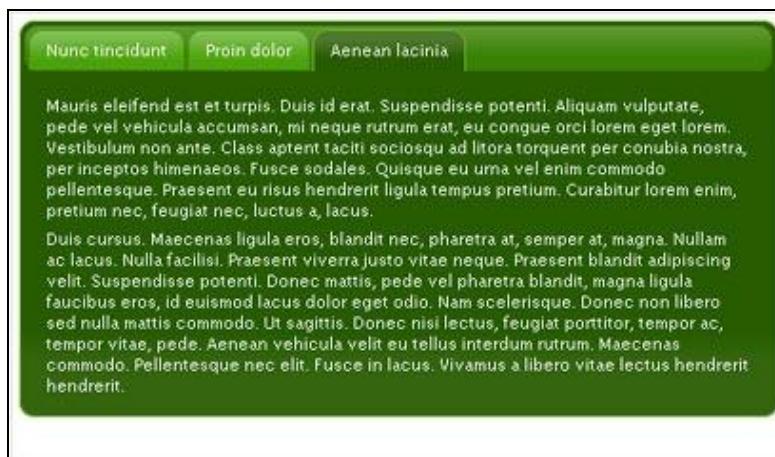


Рис. 2.24. jQueryUI, виджет Tabs

На рис. 2.24 изображен пример использования виджета Tabs, который помогает разделить информационное наполнение между несколькими вкладками. Это может быть полезно при дефиците свободного места на Web-странице.

В следующем примере покажем совместное действие виджетов Tabs и Accordion. Пример можно будет использовать как шаблон для написания большого проекта. На основе Accordion создадим главное меню из двух уровней. Заголовки — группы главного меню. В содержимое секций поместим пункты главного меню. При выборе пункта главного меню в виджете Tabs будут динамически создаваться несколько вкладок со своим содержимым. Подобный пример вы можете посмотреть на сайте <http://bazakatalogov.ru>. Файлы примера находятся на компакт-диске в папке book_primers/2-6. Для реализации примера сформируем в нашей базе данных таблицу primer_2_6_1.

Структура таблицы primer_2_6_1:

- id — первичный ключ;
- id_parent — ID родительского пункта;
- name — название пункта;
- sort — для сортировки;
- prgzag — список названий вкладок, разделенных ';' ;
- prgprg — список программ для заполнения вкладок, разделенных ';' .

Дамп для создания структуры таблицы primer_2_6_1 приведен в листинге 2.23.

Листинг 2.23

```
CREATE TABLE 'primer_2_6_1' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'id_parent' int( 11 ) default NULL ,
'name' varchar( 50 ) default NULL ,
'sort' int( 5 ) NOT NULL ,
'prgzag' varchar( 120 ) default NULL ,
'prgprg' varchar( 120 ) default NULL ,
UNIQUE KEY 'id' ( 'id' ) ,
KEY 'sort' ( 'sort' )
) ENGINE = MYISAM AUTO_INCREMENT = 135DEFAULT CHARSET = cp1251
```

Дамп базы данных находится на компакт-диске в папке book_primers/primer2-6.sql.

Загружаем таблицу primer_2_6_1 в базу данных и приступаем к программированию примера.

В файле index.php создадим только три блока — один для загрузки виджета Accordion, второй для загрузки виджета Tabs и третий для заголовка (для какого пункта Accordion будет загружать содержимое в Tabs). Содержимое файла index.php приведено в листинге 2.24. Подключаем библиотеку jQuery, файл библиотеки jQuery UI, функции хаях и файлы стилевого оформления. В примере выбрана тема sunny для jQuery UI. Вы можете выбрать иную, расскомментировав другую строку (все темы есть на компакт-диске). По событию onload документа выполним начальную загрузку главного меню — виджета Accordion.

Листинг 2.24

```
<?php
// подключение библиотеки хаях
require_once ("xajax_core/xajax.inc.php");
require_once ("create_accordion.php");
require_once ("new_tabs.php");
$xajax = new xajax();
// регистрация функций
$xajax->register(XAJAX_FUNCTION, "Create_Accordion");
```

```
$xajax->register(XAJAX_FUNCTION, "New_Tabs");

$xajax->processRequest();

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<!--<link type="text/css" href="css/ui-lightness/jquery-ui-1.7.3.custom.css"
rel="stylesheet" />
<link type="text/css" href="css/cupertino/jquery-ui-1.7.3.custom.css"
rel="stylesheet" />
<link type="text/css" href="css/pepper-grinder/jquery-ui-1.7.3.custom.css"
rel="stylesheet" />
<link type="text/css" href="css/dark-hive/jquery-ui-1.7.3.custom.css"
rel="stylesheet" />
<link type="text/css" href="css/swanky-purse/jquery-ui-1.7.3.custom.css"
rel="stylesheet" />
-->
<link type="text/css" href="css/sunny/jquery-ui-1.7.3.custom.css"
rel="stylesheet" />
<title> Пример 1 (глава 2) к книге </title>

<script type="text/javascript" src="js/jquery-1.4.2.js"></script>
<script src="js/jquery-ui-1.7.3.custom.min.js"
type="text/javascript"></script>

<?php $xajax->printJavascript(''); ?>

</head>

<body onload='xajax_Create_Accordion();'>
<!-- шапка -->
<div id=header1><b>Примеры к книге (глава 2 пример 6)<br>
    Совместное использование jQuery UI виджетов Tabs и Accordion </b></div>
<br>
<!-- -->
<table width=100%>
<tr>
<td width=20%>
    <!-- место для Accordion-->
    <div id='accordion1'></div>
</td>
<td width=5%></td>
<td width=75% style='vertical-align:top'>
```

```
<!-- заголовок -->
<div id='zag1'></div>
<!-- место для Tabs-->
<div id='alltabs1'></div>
</td>
</tr>
</table>
</body>
</html>
```

Вызываем хайк-функцию `Create_Accordion`, которая расположена в файле `createAccordion.php`. Для создания виджета `Accordion` выбором из базы данных формируется контент, приведенный в листинге 2.25.

Листинг 2.25

```
<h4><a href='#'>Рубрика 1</a></h4>
<div>
<div style='cursor:pointer' onclick='xajax_New_Tabs(4);'>Пункт 1</div>
<div style='cursor:pointer' onclick='xajax_New_Tabs(5);'>Пункт 2</div>
<div style='cursor:pointer' onclick='xajax_New_Tabs(6);'>Пункт 3</div>
</div>
<h4><a href='#'>Рубрика 2</a></h4>
<div>
<div style='cursor:pointer' onclick='xajax_New_Tabs(7);'>Пункт 4</div>
<div style='cursor:pointer' onclick='xajax_New_Tabs(8);'>Пункт 5</div>
</div>
<h4><a href='#'>Рубрика 3</a></h4>
<div>
<div style='cursor:pointer' onclick='xajax_New_Tabs(9);'>Пункт 6</div>
<div style='cursor:pointer' onclick='xajax_New_Tabs(10);'>Пункт 7</div>
</div>
```

Затем этот контент записывается в блок `div id=accordion1`. Вот js-код для запуска виджета `accordion`:

```
$('#accordion1').accordion({autoHeight:false});
```

Далее создаем виджет `Tabs` для поля **Пункт 1**. Выбором из базы данных создается контент, приведенный в листинге 2.26.

Листинг 2.26

```
<ul>
<li id='li1'><a href='#tabs1'>Вкладка1-1</a></li>;
<li id='li2'><a href='#tabs2'>Вкладка1-2</a></li>;
<li id='li3'><a href='#tabs3'>Вкладка1-3</a></li>;
```

```
<li id='li4'><a href="#tabs4">Вкладка1-4</a></li>;
<li id='li5'><a href="#tabs5">Вкладка1-5</a></li>;
</ul>
<div id='tabs1'>Результат программы для заполнения prg1-1</div>
<div id='tabs2'>Результат программы для заполнения prg1-2</div>
<div id='tabs3'>Результат программы для заполнения prg1-3</div>
<div id='tabs4'>Результат программы для заполнения prg1-4</div>
<div id='tabs5'>Результат программы для заполнения prg1-5</div>
```

Затем этот контент записывается в блок `div id=alltabs1`. Вот js-код для запуска виджета Tabs:

```
$('#alltabs1').tabs({fxFade: true, fxSpeed:'slow'})
```

Содержимое файла `create_accordion.php` приведено в листинге 2.27.

Листинг 2.27

```
<?php
function Create_Accordion()
{
    $objResponse = new xajaxResponse();
    // подключиться к базе данных
    require_once("mybaza.php");
    // сформировать контент для accordion
    $content1="";
    $query11="SELECT * FROM ".TABLE1." WHERE
              id_parent='0' ORDER BY sort ASC ";
    $rez11=mysql_query($query11);
    while($row11=mysql_fetch_assoc($rez11))
    {
        $content1.=<h4><a href='".$row11[name]."'></a></h4>;
        $query12="SELECT * FROM ".TABLE1." WHERE id_parent='".$row11[id]."'
                  ORDER BY sort ASC ";
        $rez12=mysql_query($query12);
        $content1.=<div>;
        while($row12=mysql_fetch_assoc($rez12))
        {
            $content1.=<div style='cursor:pointer'
            onclick='xajax_New_Tabs(\".$row12[id].\");'>".$row12[name]."</div>";
        }
        $content1.=</div>;
    }
    // подгрузить контент для accordion
    $objResponse->assign("accordion1","innerHTML",$content1);
    // запустить accordion
```

```

$objResponse->script("$('#accordion1').accordion({autoHeight:
    false});");

// создаем Tabs для этого пункта Accordion
// первый пункт первого меню
$query13="SELECT * FROM ".TABLE1." WHERE id_parent='1'
    ORDER BY sort ASC LIMIT 1, 2 ";
$rez13=mysql_query($query13);
$row13=mysql_fetch_assoc($rez13);
// заголовок
$objResponse->assign("zag1","innerHTML",<h3>".$row13[name]."</h3>");
$content21=<ul>".$content22="";
$arrzag=explode(";", $row13[prgzag]);
$arrprg=explode(";", $row13[prgprg]);
for ($i=1;$i<count($arrzag);$i++)
for ($i=1;$i<count($arrzag);$i++)
{
if ($arrzag[$i-1]<>"")
{
$content21.= "<li id='li".$i."'>
            <a href='#tabs".$i."'>".$arrzag[$i-1]."</a></li>";
$content22.= "<div id='tabs".$i."'>".do_prg($arrprg[$i-1])."</div>";
}
else
{
$content21.= "<li id='li".$i."'></li>";
$content22.= "<div id='tabs".$i."'></div>";
}
}
$content21.= "</ul>";
// загрузка контента для tabs
$objResponse->assign("alltabs1","innerHTML",$content21.$content22);
// запуск виджета Tabs
$objResponse->script("$('#alltabs1').tabs({fxFade: true,fxSpeed:
    'slow'});");
return $objResponse;
}

function do_prg($arg)
{
return "Результат программы для заполнения ".$arg;
}
?>

```

Результат загрузки страницы иллюстрирует рис. 2.25.

Примеры к книге (глава 2 пример 6)
Совместное использование jQuery UI виджетов Tabs и Accordion

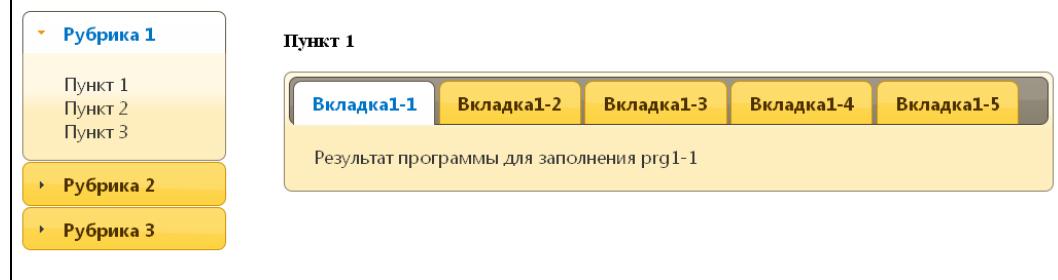


Рис. 2.25. Начальная страница примера 2-6

При выборе другого пункта меню в виджете Accordion вызывается хайперфункция New_Tabs, расположенная в файле new_tabs.php (листинг 2.28). Сначала функция очищает содержимое блока `id=alltabs1`, затем удаляет виджет Tabs, посылая js-код:

```
$('#alltabs1').tabs('destroy')
```

А далее заново создает заполнение Tabs для нового пункта меню и запускает виджет Tabs для блока `id=alltabs1`. Результат выбора другого пункта меню иллюстрирует рис. 2.26.

Листинг 2.28

```
<?php
function New_Tabs ($Id)
{
    $objResponse = new xajaxResponse();
    // очистить содержимое блока alltabs1
    $objResponse->assign("alltabs1","innerHTML","");
    // destroy tabs
    $objResponse->script("$('#alltabs1').tabs('destroy')");
    // подключиться к базе данных
    require_once("mybaza.php");
    // найти пункт меню
    $query13="SELECT * FROM ".TABLE1." WHERE id='".$Id."'";
    $rez13=mysql_query($query13);
    $row13=mysql_fetch_assoc($rez13);
    // заголовок
    $objResponse->assign("zag1","innerHTML",<h3>".$row13[name].</h3>");
    // список названий и программ
    $arrzag=explode(";", $row13[prgzag]);
    $arrprg=explode(";", $row13[prgprg]);
    $content21=<ul>;
```

```

for($i=1;$i<count($arrzag);$i++)
{
if($arrzag[$i-1]<>"")
{
$content21.= "<li id='li".$i."'><a href='#tabs".$i."'>
".$arrzag[$i-1]."</a></li>";
$content22.= "<div id='tabs".$i."'>".do_prg($arrprg[$i-1])."</div>";
}
else
{
$content21.= "<li id='li".$i."'></li>";
$content22.= "<div id='tabs".$i."'></div>";
}
}

$content21.= "</ul>";
// новый контент в блок alltabs1
$objResponse->assign("alltabs1","innerHTML",$content21.$content22);
// запустить tabs
$objResponse->script("$('#alltabs1').tabs({fxFade: true,fxSpeed:
'slow'})");
return $objResponse;
}
?>

```

Примеры к книге (глава 2 пример 6)
Совместное использование jQuery UI виджетов Tabs и Accordion

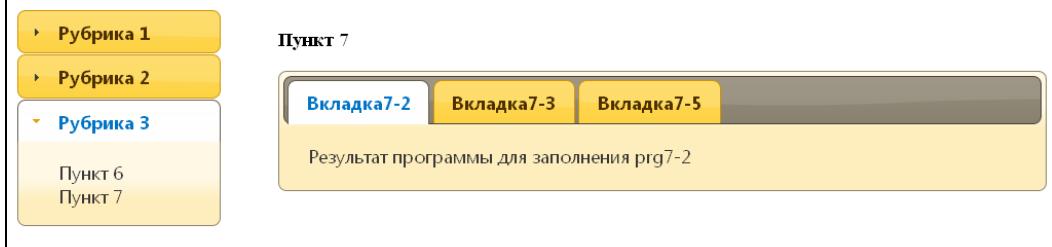


Рис. 2.26. Выбор другого пункта меню

Заполнение контента вкладок Tabs происходит следующим образом. Каждой вкладке в базе данных для каждого пункта меню соответствует свое название программы, которое передается в функцию `do_prg`, расположенную в файле `create_accordion.php`. В нашем примере она просто возвращает строку **Результат программы для заполнения...** Здесь вы можете записать свой обработчик (при мер приведен в листинге 2.29).

Листинг 2.29

```
function do_prg($arg)
{
switch ($arg)
{
case prg1-1: $result=f11();
break;

case prg5-5: $result=f55();
break;

}

$result;
}
```

2.4.3.3. Галерея товаров с формой заказа

Создадим в качестве примера небольшой скрипт магазина с возможностью выбора языка (английский и русский) — продажа коллекций gif-галерей с возможностью просмотра всех фото каждой галереи, с формой заказа. Будем использовать следующие плагины: imageFlow (для просмотра галереи рисунков), fancybox (для создания формы заказа). Меню реализуем средствами базовой библиотеки jQuery. С помощью библиотеки хаах осуществим подгрузку контента страниц "О сайте" и "Контакты". Файлы примера находятся на компакт-диске в папке book_primers/2-7. Страница примера при загрузке приведена на рис. 2.27.

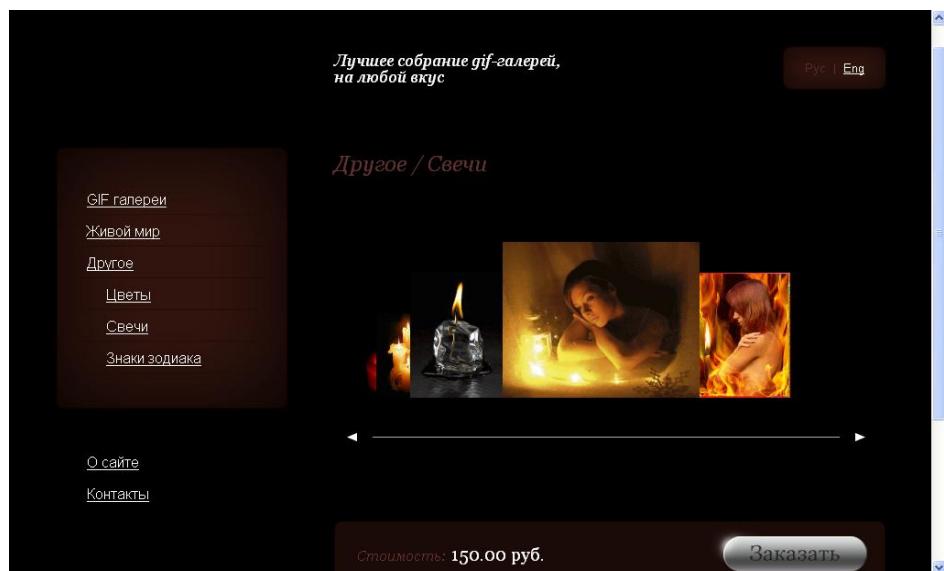


Рис. 2.27. Вид магазина при загрузке

Для нашего примера создадим шесть таблиц в базе данных:

- primer_2_7_kategory — список категорий товаров;
- primer_2_7_tovars — список товаров;
- primer_2_7_images — список картинок для каждого товара;
- primer_2_7_content — содержимое для страниц "О сайте", "Контакты";
- primer_2_7_rate — курс валют;
- primer_2_7_zakaz — учет заказов.

Дампы для создания структуры таблиц базы данных приведены в листинге 2.30 (дамп находится на компакт-диске в папке book_primers/primer2-7.sql).

Листинг 2.30

```

CREATE TABLE 'book_xajax1'.primer_2_7_kategory' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'name_rus' varchar( 20 ) default NULL ,
'name_eng' varchar( 20 ) default NULL ,
'sort' int( 5 ) NOT NULL ,
'visible' set( 'yes', 'no' ) NOT NULL default 'yes',
UNIQUE KEY 'id' ( 'id' ) ,
KEY 'sort' ( 'sort' )

) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251

CREATE TABLE 'book_xajax1'.primer_2_7_tovars' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'name_rus' varchar( 20 ) default NULL ,
'name_eng' varchar( 20 ) default NULL ,
'id_kategory' int( 9 ) NOT NULL ,
'sort' int( 5 ) NOT NULL ,
'info_rus' text,
'info_eng' text,
'pay_rus' float( 10, 2 ) default NULL ,
'pay_eng' float( 5, 2 ) default NULL ,
'visible' set( 'yes', 'no' ) NOT NULL default 'yes',
UNIQUE KEY 'id' ( 'id' )

) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251

CREATE TABLE 'book_xajax1'.primer_2_7_images' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'id_tovar' int( 9 ) NOT NULL ,
'name' varchar( 40 ) default NULL ,
'sort' int( 5 ) NOT NULL ,
'info_rus' varchar( 20 ) NOT NULL ,
'info_eng' varchar( 20 ) NOT NULL ,
'visible' set( 'yes', 'no' ) NOT NULL default 'yes',
UNIQUE KEY 'id' ( 'id' )

```

```
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
CREATE TABLE 'book_xajax1'.'primer_2_7_content' (
'id' int( 2 ) NOT NULL AUTO_INCREMENT ,
'zag_rus' varchar( 20 ) NOT NULL ,
'zag_eng' varchar( 20 ) NOT NULL ,
'info_rus' text NOT NULL ,
'info_eng' text NOT NULL ,
UNIQUE KEY 'id' ( 'id' )

) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
CREATE TABLE 'book_xajax1'.'primer_2_7_rate' (
'id' int( 5 ) NOT NULL AUTO_INCREMENT ,
'data' date NOT NULL ,
'usd' float( 8, 4 ) default NULL ,
UNIQUE KEY 'id' ( 'id' )

) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
CREATE TABLE 'book_xajax1'.'primer_2_7_zakaz' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'data_zakaz' datetime NOT NULL ,
'data_oplata' date NOT NULL ,
'data_dostavka' date NOT NULL ,
'id_tovar' int( 9 ) NOT NULL ,
'name' varchar( 20 ) NOT NULL ,
'phone' varchar( 15 ) NOT NULL ,
'email' varchar( 30 ) NOT NULL ,
'address' varchar( 100 ) NOT NULL ,
'oplata' set( 'yes', 'no' ) NOT NULL default 'no',
'dostavka' set( 'yes', 'no' ) NOT NULL default 'no',
'visible' set( 'yes', 'no' ) NOT NULL default 'yes',
UNIQUE KEY 'id' ( 'id' )

) ENGINE = MYISAM DEFAULT CHARSET = cp1251;
```

Так как пример двуязычный, в таблицах primer_2_7_kategory, primer_2_7_tovars и primer_2_7_content предусмотрены поля для двух языков. В таблице primer_2_7_rate хранятся курсы валют для пересчета цены в доллары. В таблице primer_2_7_images находятся картинки для каждой галереи. В таблицу primer_2_7_zakaz будем записывать данные заказов. Начальные данные для вывода начального товара и языка загрузки хранятся в файле my.php. При загрузке файла index.php подключаются файлы библиотек, анализируются данные HTTP-запроса и выводится информация о нужном товаре, по окончании загрузки страницы запускаются плагины. Для страниц "О сайте" и "Контакты" вызываются хайх-функции, изменяющие контент страницы. Содержимое файла index.php приведено в листинге 2.31.

Листинг 2.31

```
<?php
require_once('index.common.php');
require_once("mybaza.php");
require_once("my.php");
// язык
$query0="SELECT usd FROM ".BAZA_RATE." ORDER BY id DESC LIMIT 0, 1 ";
$_SESSION[usd]=mysql_result(mysql_query($query0),0);
if(!isset($_GET[prg]))
    $_SESSION[language]=LANGUAGE;
else
    $_SESSION[language]=$_GET[language];
// товар
if(!isset($_GET[tovar]))
    $_SESSION[tovar]=TOVAR_ID;
else
    $_SESSION[tovar]=$_GET[tovar];
// программа - tovar,about, contacts
if(!isset($_GET[prg]))
    $_SESSION[prg]='tovar';
else
    $_SESSION[prg]=$_GET[prg];
// если tovar -
if($_SESSION[prg]=='tovar')
{
    // товар для вывода
    $query11="SELECT * FROM ".BAZA_TOVAR." WHERE id=".$_SESSION[tovar]." ";
    $rez11=mysql_query($query11);
    $row11=mysql_fetch_assoc($rez11);
    // категория начального товара
    $query12="SELECT * FROM ".BAZA_CATEGORY." WHERE
        id=".$row11[id_kategory].";
    $rez12=mysql_query($query12);
    $row12=mysql_fetch_assoc($rez12);
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<meta http-equiv="Content-Type" charset=windows-1251" >
<title>Title</title>
<link rel="stylesheet" type="text/css" href="css/reset.css" >
<link rel="stylesheet" type="text/css" href="css/typography.css">
```

```
<link rel="stylesheet" type="text/css" href="css/style.css" >
<style type="text/css">
    img, div { behavior: url(iepngfix.htc) }
</style>
<link rel="stylesheet" href="js/fancybox/jquery.fancybox-1.3.0.css"
type="text/css" media="screen">
<script type="text/javascript" src="iepngfix_tilebg.js"></script>
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/fancybox/jquery.fancybox-
1.3.0.js"></script>
<script type="text/javascript">
$(document).ready(function(){
$('.sub + ul').hide();
$('.sub').toggle(
function(){$(this).next('ul').slideDown(500); },
function(){$(this).next('ul').slideUp(500); }
);
<?php
// открыть нужную категорию
if($_SESSION[prg]=='tovar')
echo '$("#ul'.$row12[id].')."slideDown(500),';
?>
$("a#inline").fancybox({
'overlayColor': '#000',
'overlayOpacity': 0.7,
'showCloseButton': false,
'padding': 0,
'margin': 0,
'titleShow': false,
'showNavArrows': false,
'width':651,
'height':402,
'autoDimensions':false
});
});
</script>
<link rel="stylesheet" href="imageflow.packed.css" type="text/css" />
<script type="text/javascript" src="imageflow.js"></script>
<?php $xajax->printJavascript(''); ?>
</head>
<?php
if($_SESSION[prg]=='tovar')
echo "<body>";
// для О сайте и Контакты - вызов хаях-функций
elseif($_SESSION[prg]=='contacts')
```

```
echo "<body onload='xajax_Contacts();'>";
else
    echo "<body onload='xajax_About();'>";
?>
<div id="container">
<div id="header">
<div id="sl">
<?php
$query14="SELECT info_".$_SESSION[language]." FROM ".BAZA_CONTENT." WHERE id=3";
$rez14=mysql_query($query14);
$slogan1="";
$slogan1=<p id='slogan1'>".mysql_result($rez14,0)."</p>";
echo $slogan1;
?>
</div>
<div id="lang">
<?php
$lang="";
if($_SESSION[language]=='rus')
    $lang=<span>Pyc &nbsp; | &nbsp; <a href='index.php?language=eng&prg='.
        $_SESSION[prg]."&tovar=". $_SESSION[tovar]."'>Eng</a></span>";
else
    $lang=<span><a href='index.php?language=rus&prg='.
        $_SESSION[prg]."&tovar=". $_SESSION[tovar]."'>
        Pyc</a> | &nbsp; Eng &nbsp;</span>";
echo $lang;
?>
</div>
</div>
<div id="side">
    <div id="menu">
        <div id="menutopbg">
            <div id="menubotbg">
                <?php
                    // Вывод категорий и товаров
                    $menu1="";
                    $menu1.= "<ul class='menuul'>";
                    if($_SESSION[language]=='rus')
                        $menu1.= "<li><a class='mitem' href='#'>GIF галереи</a></li>";
                    else
                        $menu1.= "<li><a class='mitem' href='#'>GIF gallery</a></li>";
                    $query21="SELECT * FROM ".BAZA_KATEGORY." WHERE visible='yes'
                            ORDER BY sort ASC ";
                    $rez21=mysql_query($query21);
                <?php
            </div>
        </div>
    </div>
</div>
```

```
while($row21=mysql_fetch_assoc($rez21))
{
$menu1."<li><a class='mitem sub' href='#'>".
$row21['name_'].$_SESSION[language]."</a>" ;
$menu1."<ul id='ul".$row21[id]."' class='submenu'>" ;
$query22="SELECT * FROM ".BAZA_TOVAR." WHERE
id_kategory=".$row21[id]." && visible='yes'
ORDER BY sort ASC ";
$rez22=mysql_query($query22);
while($row22=mysql_fetch_assoc($rez22))
{
$menu1."<li><a class='smitem' href='index.php?language=". $_SESSION[language]."&pрг=tovar&tovar='".$row22[id]."'>".
$menu1.">".$row22['name_'].$_SESSION[language]."</a></li>" ;
}
$menu1."</ul>";
$menu1."</li>";
}
$menu1."</ul>";
echo $menu1;
?>
</div>
</div>
</div>
<div id=menu2>
<?php
// вывод О сайте и Контакты
$menu2=<ul class='menuul'>";
if($_SESSION[language]=='rus')
{
$menu2."<li class='mitem nobg'><a href='index.php?language=rus&pрг=about&tovar=".$_SESSION[tovar]."' class='nobg'>О сайте</a></li>" ;
$menu2."<li class='mitem nobg'><a href='index.php?language=rus&pрг=contacts&tovar=".$_SESSION[tovar]."' class='nobg'>Контакты</a></li>" ;
}
else
{
$menu2."<li class='mitem nobg'><a href='index.php?language=eng&pрг=about&tovar=".$_SESSION[tovar]."' class='nobg'>About</a></li>" ;
$menu2."<li class='mitem nobg'><a href='index.php?language=eng&pрг=contacts&tovar=".$_SESSION[tovar]."' class='nobg'>Contacts</a></li>" ;
}
$menu2."</ul>";
echo $menu2;
```

```
?>
</div>
</div>
<div id="main">
<div id='headtovar'>
<?php
// заголовок товара
if($_SESSION[prg]=='tovar')
{
$zag="";
$zag.="

## ".$row12['name_'].$_SESSION[language]."/ "; $zag.= $row11['name_'].$_SESSION[language]."</h2>"; } elseif($_SESSION[prg]=='contacts') { if($_SESSION[language]=='rus') $zag=" Контакты </h2>"; else $zag=" Contacts </h2>"; } else { if($_SESSION[language]=='rus') $zag=" О сайте </h2>"; else $zag=" About </h2>"; } echo $zag; ?> </div> <div id="myImageFlow" class="imageflow"> <?php // фотографии товара $image=""; $query13="SELECT * FROM ".BAZA_IMAGES." WHERE id_tovar='".$_SESSION[tovar]."' ORDER BY sort ASC"; $rez13=mysql_query($query13); while($row13=mysql_fetch_assoc($rez13)) { $image.= "<img src='img/".$row13[name]."'" . " "; $image.= "longdesc='img/".$row13[name]."'" . " "; $image.= "alt='".$row13[name]."' />"; } if($_SESSION[prg]=='tovar') echo $image;


```

```
?>
</div>
<div id="opisanie" class="opisanie">
<?php
if($_SESSION[prg]=='tovar')
{
    $opisanie="";
    $opisanie.= $row11['info_'].$_SESSION[language];
    echo $opisanie;
}
?>
</div>
<br>
<div id="allprice">
<?php
if($_SESSION[language]=='rus')
{
    $price=<div id='price'>";
    $price.= "<span id='spanprice'>Стоимость: <span>".$row11[pay_rus]."
                руб.</span></span>";
    $price.= "<a id='inline' href='#data'></a>";
    $price.= "</div>";
}
else
{
    $price=<div id='price'>";
    $price.= "<span>Pay: <span>".sprintf('%5.0f',
        ($row11[pay_rus]/$_SESSION[usd])). " $</span></span>";
    $price.= "<a id='inline' href='#data'></a>";
    $price.= "</div>";
}
if($_SESSION[prg]=='tovar')
    echo $price;
else
    echo "<table></table>";
?>
</div>
<div id="footer"><span class="left"></span><span class="right"></span></div>
</div>
<div class="clear"></div>
</div>
<div style="display: none; ;">
<div id="data">
<?php
$tovar=$row12['name '.$_SESSION[language]]." /
".$row11['name_'].$_SESSION[language];
```

```
if($_SESSION[language]=='rus')
{
$name="Ваше имя";
$phone="Контактный <br/> телефон";
$email="E-mail";
$address="Адрес доставки";
}
else
{
$name="Your name";
$phone="Contact <br/> phone";
$email="E-mail";
$address="Address";
}

$data=<form id='Form_Zakaz' action='javascript:void(null);' onsubmit='
    xajax_Create_Zakaz(xajax.getFormValues(\"Form_Zakaz\"));'>
<table id='buy'>
<tr><td></td><td><h3>".$tovar."</h3></td></tr>
<tr><td class='right'>".$name."</td><td><input name='name' type='text'>
<input name='tovar' value='".$SESSION[tovar]."' type='hidden'></td></tr>
<tr><td class='right'>".$phone."</td><td>
+<input name='phone1' id='phone1' type='text' size=2 maxlength=2
value=''
(<input name='phone2' id='phone2' type='text' value='' size=5 maxlength=5)
<input name='phone3' id='phone3' type='text' value='' size=7 max-
length=7)</td></tr>
<tr><td class='right'>".$email."</td><td><input name='email' id='email'
type='text' value=''/></td></tr>
<tr><td class='right'>".$address."</td><td><textarea name='address' cols='40'
rows='3'></textarea></tr>
<tr><td></td><td><input name='submit' type='image' src='img/submit.png'
id='submit'></td></tr>
</table>
</form>";
echo $data;
?>
</div>
</div>
<div id="big_img" style='position: absolute; z-index: 299999900;
margin: 20px; top: 5%; left: 5%; opacity: 1.0;
width: 90%; height: auto; display: none;
background-color: rgb(0, 0, 0)' onclick='this.style.display="none";'
onmouseout='this.style.display="none";'>
</div>
</body>
</html>
```

Вид страницы при выборе другого языка иллюстрирует рис. 2.28. Содержимое файла `create_zakaz.php` приведено в листинге 2.33. Плагин `imageFlow` позволяет организовать галерею для просмотра картинок. Выбранная картинка перемещается в центр, увеличивается в размере и приобретает фокус. Соседние картинки уходят на второй план и "замыливаются". При щелчке кнопкой мыши по картинке в центре блок `div` с `id=big_img` становится видимым, и в его содержимое передается увеличенное изображение картинки. Для этого в стандартном файле `imageflow.js` (листинг 2.32) внесем изменения в свойство `click` объекта `imageflow` (листинг 2.33).

Листинг 2.32

```
/*onClick: function() { document.location = this.url; }, */
/* Onclick behaviour */
onClick: function()
{
    document.getElementById('big_img').style.display='block';
    document.getElementById('big_img').innerHTML=
    ' '>,
```

Вид страницы при просмотре увеличенной картинки изображен на рис. 2.29. Картинка пропадает при щелчке на ней мышью либо при выходе мыши за пределы блока:

```
onclick='this.style.display="none";'
onmouseout='this.style.display="none";'
```

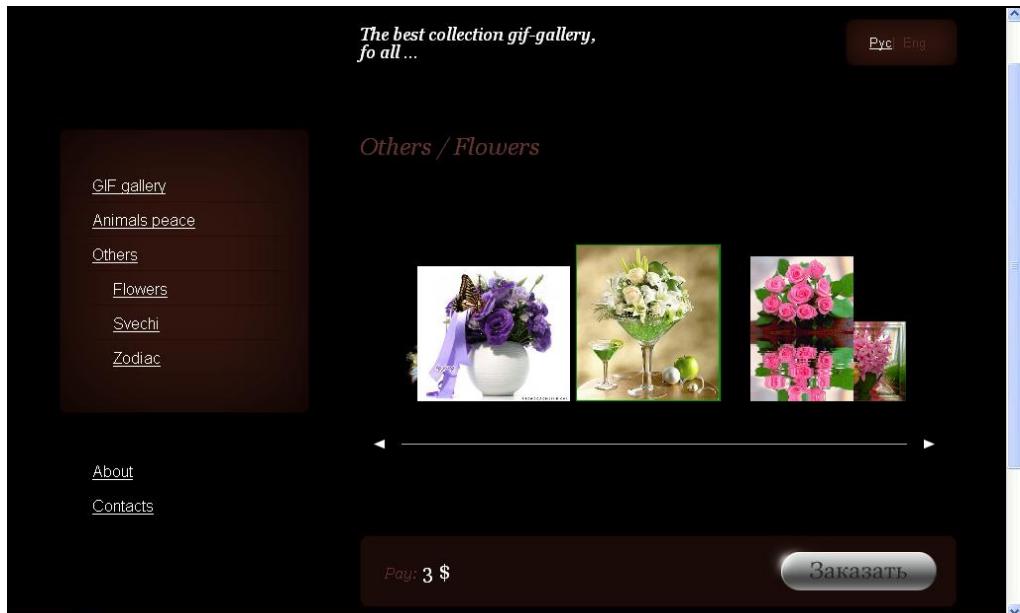


Рис. 2.28. Вид страницы при изменении языка

При нажатии на кнопку **Заказать** появляется форма заказа (рис. 2.30) — еще одно применение плагина fancybox. При заполнении заказа и нажатии на кнопку **Подтвердить заказ** вызывается хук-функция `Create_Zakaz`, расположенная в файле `create_zakaz.php` (листинг 2.33). Функция проверяет правильность заполнения формы и записывает информацию о заказе в базу данных и отправляет сообщение о поступившем заказе на e-mail администратора.



Рис. 2.29. Увеличенное изображение картинки

The best collection gif-gallery,
fo all... Рус Eng

Animals peace / Cats

Your name

Contact phone + ()

E-mail

Address

 PO!

[GIF gallery](#)
[Animals peace](#)
[Cats](#)
[Babochki](#)
[Fishes](#)
[Others](#)

[About](#)
[Contacts](#)

Pay 5 \$

Рис. 2.30. Вид формы заказа

Листинг 2.33

```
<?php
function Create_Zakaz($Id)
{
    $objResponse = new xajaxResponse();
    // подключение к базе данных
    require_once("mybaza.php");
    // подключение файла настроек
    require_once("my.php");
    // получение данных
    $tovar=$Id[tovar];
    $name=utftowin($Id[name]);
    $phone1=$Id[phone1];
    $phone2=$Id[phone2];
    $phone3=$Id[phone3];
    $email=$Id[email];
    $address=utftowin($Id[address]);
    // ***** проверка ****
    $error="";
    // name
    if(!ereg("^[a-z,A-Z,а-я,А-Я,0-9,\ ]{6,50}$", $name))
    {
        if($_SESSION[language]=='rus')
            $error.=" Неверный формат имени ";
        else
            $error.=" Неверный формат name ";
    }
    // phone
    if(!ereg("^[0-9,\+\(\)\]{1,2}$", $phone1))
    {
        if($_SESSION[language]=='rus')
            $error.=" Неверный формат телефон (код страны) ";
        else
            $error.=" Неверный формат phone (код страны) ";
    }
    if(!ereg("^[0-9,\+\(\)\]{3,5}$", $phone2))
    {
        if($_SESSION[language]=='rus')
            $error.=" Неверный формат телефон (код города) ";
        else
            $error.=" Неверный формат phone (код town) ";
    }
    if(!ereg("^[0-9,\+\(\)\]{5,7}$", $phone3))
```

```

{
if($_SESSION[language]=='rus')
    $error.=" Неверный формат телефон (номер)";
else
    $error.=" Неверный формат phone (number)";
}
// email
if(!ereg("^( [a-z,0-9,-\.] {2,20}) ([\@]{1}) ([a-z,0-9,-_]{2,20})"
([\.]{1}) ([a-z,]{1,3}) $", $email))
{
if($_SESSION[language]=='rus')
    $error.=" Неверный формат e-mail ";
else
    $error.=" Неверный формат e-mail ";
}
if(strlen(trim($address))==0)
{
if($_SESSION[language]=='rus')
    $error.=" Введите адрес ";
else
    $error.=" Введите address ";
}
// ошибка
if($error<>"")
{
$objResponse->alert($error);
return $objResponse;
}
// **** норма - отправка e-mail админу и запись в базу ****
// запись в базу
$phone= ".$phone1." ".$phone2.". ".$phone3;
$data_zakaz=date('Y-m-d H:i:s');
$query1="INSERT INTO ".BAZA_ZAKAZ." SET data_zakaz='".$data_zakaz."',
id_tovar='".$tovar."', name='".$name."',
phone='".$phone."', email='".$email."',
address='".$address."' ";
$rez1=mysql_query($query1);
$num=mysql_insert_id();
// формирование контента
$content="";
$content.= "<table id='buy'><td><h3>";
if($_SESSION[language]=='rus')
    $content.= "<center><br><br><br> <center>Ваш заказ
<br>принят <br> Номер заказа ".$num."</center>";
else

```

```
$content.=<center><br><br><br> Your zakaz <br>принят <br>Number  
zakaz ".$num." </center>";  
$content.= "</h3></td></tr></table>";  
// выдача контента  
$objResponse->assign("data","innerHTML",$content);  
// отправка письма администратору  
$query2="SELECT kategory.name_rus,tovar.name_rus FROM  
kategory,tovar WHERE tovar.id=\"$tovar.\" &&  
kategory.id=tovar.id_kategory ";  
$rez2=mysql_query($query2);  
$row2=mysql_fetch_row($rez2);  
$to=EMAILADMIN;  
$subject='Заказ с сайта'.SITE;  
$body='<b>Заказ на сайте '.SITE.'</b><br>';  
$body.='<b>Заказан следующий товар</b><br>';  
$body.='<b>'.$row2[0].' \ '. $row2[1].'</b><br><br>';  
$body.='<b>Данные покупателя:</b><br>';  
$body.='<b>Имя – </b>'.$name.'<br>';  
$body.='<b>Контактный телефон – </b>'.$phone.'<br>';  
$body.='<b>E-mail – </b>'.$email.'<br>';  
$body.='<b>Адрес – </b>'.$address.'<br>';  
$body.='<b>Дата заказа – </b>'.date('d-m-Y H:i:s').'<br>';  
$body.='<b>Номер заказа – </b>'.$num.'<br>';  
$body.='<br>';  
$headers="Content-type: text/html; charset=windows-1251;";  
mail($to,$subject,$body,$headers);  
return $objResponse;  
}  
?>
```

2.5. Хайк и Smarty

2.5.1. Что такое Smarty

Smarty — шаблонный движок для PHP, позволяющий разделить прикладную логику, содержание и представление. Это очень удобно в ситуациях, когда программист и проектировщик шаблона играют различные роли (или это различные люди). Например, вы создаете страницу, которая показывает газетную статью. Название статьи, автор и сама статья — элементы, которые не содержат никакой информации о том, как они будут представлены. Они передаются в Smarty-шаблон, а верстальщик редактирует шаблоны и использует комбинацию тегов HTML и шаблона, чтобы отформатировать представление этих элементов (таблицы HTML, фоновые цвета, размеры шрифта, стиля и т. д.). Однажды программист захочет изменить способ хранения статьи (внести изменения в логику приложения).

Это не затрагивает проектировщика шаблонов. Содержание будет все еще передаваться в шаблон таким же самым способом. Аналогично, если проектировщик хочет полностью перепроектировать шаблоны, то не потребуется никаких изменений в прикладной логике. Поэтому программист может менять прикладную логику, не затрагивая шаблоны, а проектировщик шаблона может корректировать шаблоны без привязки к прикладной логике.

Конечно, шаблоны могут содержать в себе логику, но лишь при условии, что эта логика необходима для правильного представления данных. Такие задачи, как подключение других шаблонов, чередующаяся окраска строчек в таблице, приведение букв к верхнему регистру, циклический проход по массиву для его отображения и т. д. — все это примеры логики представления. Не следует думать, что Smarty заставляет вас разделять логику приложения и представление. Smarty "не видит" разницы между этими вещами, так что помещать или не помещать логику приложения в шаблоны — решать вам. Если же вы считаете, что в шаблоне вообще не должно быть логики, то можете ограничиться использованием чистого текста и переменных.

Некоторые особенности Smarty:

- Он очень быстр.
- Он эффективен, так как обработчик PHP делает за него "грязную" работу.
- Никакой лишней обработки шаблонов, они компилируются только один раз.
- Перекомпилируются только те шаблоны, которые изменились.
- Вы можете создавать пользовательские функции и модификаторы, что делает язык шаблонов чрезвычайно расширяемым.
- Предусмотрены настраиваемые разделители тегов шаблона, например {}, {{}}, <!-- { } --> и т. д.
- Конструкции if/elseif/else/endif передаются обработчику PHP, так что синтаксис выражения {if ...} может быть сколь угодно простым (или сложным).
- Допустимо неограниченное вложение секций, условий и т. д.
- Существует возможность включения PHP-кода прямо в ваш шаблон, однако обычно в этом нет необходимости (и это не рекомендуется), т. к. движок весьма гибок и расширяем.
- Встроенный механизм кэширования.
- Произвольные источники шаблонов.
- Пользовательские функции кэширования.
- Компонентная архитектура.

2.5.2. Установка Smarty

Для получения библиотеки Smarty с официального сайта, перейдите по адресу <http://www.smarty.net/download.php>. В появившемся окне (рис. 2.31) выберите версию, например последнюю стабильную (Latest Stable Release 2.6.26) и нажмите

на ссылку **zip** (рис. 2.31). Сохраните файл на компьютере. Распакуйте архив в корневой каталог сайта.

Рассмотрим пример использования Smarty (листинг 2.34).

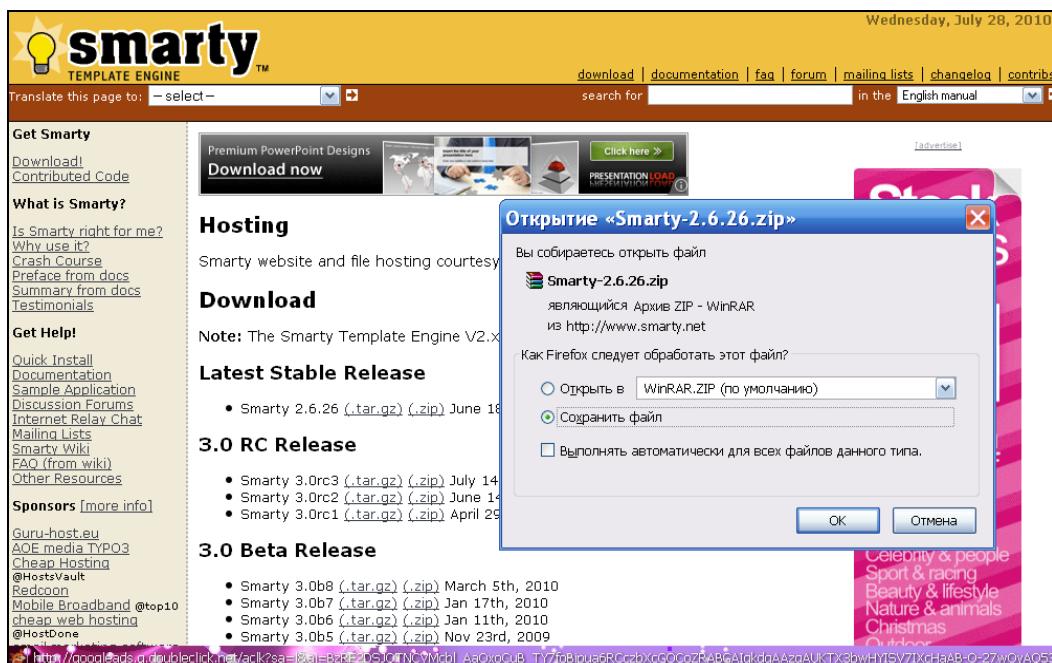


Рис. 2.31. Скачивание библиотеки Smarty

Листинг 2.34

```
// index.php
<?php
    // подключаем файл с описанием класса
    require_once("Smarty/libs/Smarty.class.php");
    // создание нового класса
    $smarty = new Smarty();
    // путь к каталогу шаблонов
    $smarty->template_dir = 'my_templates';
    $smarty->compile_dir = 'my_templates_c';

    $value1="Заголовок";
    $value2="Текст Текст Текст Текст Текст ";
    // присвоить значения переменным Smarty
    $smarty->assign("zag",$value1);
    $smarty->assign("txt",$value2);
```

```
// выводим обработанный шаблон
$smarty->display("index.tpl.html");
?>
// index.tpl.html
Мой сайт
<b>{$zag}</b>
<br>
{$txt}
```

При запуске файла index.php будет выведено следующее сообщение:

Заголовок1

Текст Текст Текст Текст Текст

2.5.3. Синтаксис шаблонов Smarty

Smarty не просто представляет собой класс для обработки шаблонов, он определяет целый язык построения шаблонов. Мы коснемся только основных его элементов. Итак, что представляет собой шаблон Smarty? Это набор специальных конструкций (переменных, вызовов функций и методов и т. п.) и HTML-тегов. Все элементы (теги) языка шаблонов Smarty заключаются между символами-ограничителями. По умолчанию это фигурные скобки { и }, но их можно изменить. Все, что не заключено в такие ограничители, Smarty рассматривает как константы, не требующие обработки. В шаблоне index.tpl.html (листинг 2.34) {\$zag} и {\$txt} — это переменные, а строка Мой сайт — константа, не изменяющаяся в процессе обработки шаблона.

Комментарии в Smarty записываются между двумя звездочками:

```
{* Это комментарий. После обработки шаблона
он на экране не отображается *}
```

Каждый Smarty-тег либо выводит значение переменной, либо вызывает какую-либо функцию.

Синтаксис записи функции:

```
{имя_функции атрибут1="значение1"
    атрибут2="значение2"}
```

Переменные в шаблоне могут быть нескольких типов:

- Переменные, значение которым присваивается в PHP-скрипте пользователя, должны иметь перед именем знак доллара:
{\$first_name}
- Элементы массива, значения которых были присвоены в PHP-скрипте пользователя, доступны в шаблоне с помощью синтаксиса {\$имя_массива.ассоциативный_ключ}:
{\$person.last_name}
- Элементы не ассоциативного массива доступны с помощью синтаксиса квадратных скобок {\$имя_массива[числовой_индекс]}:
\$person[2]}

- Свойства объектов, заданные в PHP-скрипте, доступны в шаблоне с помощью синтаксиса `{$имя_объекта->имя_свойства}:`
 `{$person->email}`
- Переменные, загруженные из конфигурационных файлов (что это такое, мы расскажем чуть позже), заключаются между символами `#`. Также они доступны как элементы ассоциативного массива `$smarty.config`:
`{#bodyBgColor#}` или `${$smarty.config.bodyBgColor}`
- Существует переменная `{$smarty}`, зарезервированная для некоторых специальных переменных шаблона (HTTP-запрос, дата и время и т. п.).

В шаблонах Smarty определен ряд модификаторов, которые можно применять к переменным, пользовательским функциям или строкам с тем, чтобы модифицировать их значения. Чтобы применить модификатор, нужно указать его название после вертикальной черты, следующей за именем переменной, функции или строкой, к которой он применяется.

Например, чтобы перевести значение переменной `{$title}` в верхний регистр, нужно применить к ней модификатор `upper`, т. е. написать следующее: `{$title|upper}`.

Можно указать сразу несколько модификаторов, отделяя их друг от друга прямой вертикальной чертой. Например, `{$title|upper|truncate}` переведет значение переменной в верхний регистр и урежет до 80 символов.

Перечислять все имеющиеся модификаторы мы не будем. Их список можно найти в документации Smarty. Скажем только, что с их помощью можно посчитать число символов, слов и параграфов, дописать строку, задать формат вывода даты и времени, сделать регулярную замену и многое другое.

2.5.4. Методы класса Smarty

Для работы с шаблонами класс Smarty определяет набор методов. Рассмотрим несколько основных методов, которые нам пригодятся для примера совместного использования хаях и Smarty.

2.5.4.1. Метод `assign`

Синтаксис:

```
void assign (смешанное значение);  
void assign (имя переменной,  
            смешанное значение);
```

Метод служит для присваивания значения переменным шаблона (листинг 2.35).

Листинг 2.35

```
<?php  
// передаем значение для  
// переменной Name  
$smarty->assign ("Name", "Иван");  
// таким образом, переменная Name  
// получит соответствующее значение Иван  
?>
```

2.5.4.2. Метод *display*

Синтаксис:

```
void display(путь к файлу шаблона);
```

Метод отображает шаблон.

Пример

```
<?php
$smarty->display("template1.tpl");
?>
```

2.5.4.3. Метод *fetch*

Синтаксис:

```
string fetch(шаблон);
```

Этот метод возвращает обработанный шаблон в строковую переменную вместо того, чтобы выводить его на экран.

Пример

```
<?php
$x=$smarty->fetch("template1.tpl");
?>
```

2.5.5. Использование хајах и Smarty

А сейчас приведем пример использования хајах в шаблонном движке Smarty. Рассмотрим два варианта применения шаблонов Smarty:

- формирование главной страницы;
- динамическую подгрузку результатов запросов к серверу.

Задействуем базу данных КЛАДР, знакомую нам по примеру из разд. 2.3.2. При выборе региона будем отображать 10 первых районов. Данные будем выводить в блок результатов, используя разные файлы шаблонов (файл шаблона выбирается здесь же в форме). Файлы примера расположены на прилагаемом компакт-диске в папке book_primers/2-8. Вид страницы при открытии файла index.php (листинг 2.36) приведен на рис. 2.32. Для вывода хајах-функций в заголовок шаблона подставляется код:

```
{$xajax_javascript}
```

Следующий код создает эту переменную в файле index.php:

```
$smarty->assign("xajax_javascript", $xajax->getJavascript("."));
```

Страница выводится через шаблон my_templates/index.tpl.html (листинг 2.37).

Примеры к книге (глава 2 пример 8)
Использование Smarty при динамической подгрузке контента



Рис. 2.32. Главная страница примера 2-8

Листинг 2.36

```
<?php
    // подключение библиотеки Smarty
    require_once("Smarty/libs/Smarty.class.php");
    // создание нового экземпляра класса
    $smarty = new Smarty();
    // путь к папке шаблонов
    $smarty->template_dir = 'my_templates';
    $smarty->compile_dir = 'my_templates_c';
    // подключение библиотеки хаjax
    require_once ("xajax_core/xajax.inc.php");
    // включение файлов
    require_once ("result_select.php");
    $xajax = new xajax();
    // регистрация функций
    $xajax->register(XAJAX_FUNCTION,"Result_Select");
    // запуск процесса
    $xajax->processRequest();
    // подключение к базе данных
    require_once("mybaza.php");
    // создание переменной – массива результатов
    $query1="SELECT id,name,socr FROM ".TABLE1." WHERE id_rayon=0 &&
        id_town=0 && id_punkt=0 ORDER BY id ASC";
    $rez1=mysql_query($query1);
    $i=0;
    while($row1=mysql_fetch_assoc($rez1))
    {
        $i++;
        $info[$i][id]=$row1[id];
        $info[$i][name]=$row1[name];
        $info[$i][socr]=$row1[socr];
    }
```

```
// создание переменной для хаях-функций
$smarty->assign("xajax_javascript", $xajax->getJavascript("."));
// создание переменной для результата
$smarty->assign("info", $info);
// вывод шаблона
$smarty->display("index.tpl.html");
?>
```

Листинг 2.37

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title> Пример 8 (глава 2) к книге </title>
{$xajax_javascript}
</head>
<body>
<!-- шапка -->
<div id=header1><b>Примеры к книге (глава 2 пример 8)<br>
Использование Smarty при динамической подгрузке контента</b></div>
<br>
<!-- Форма -->
<form id='FormSelectRegion' method='post' action='javascript:void(null);'
      onsubmit='
        xajax.$("ButtonFormSelectRegion").disabled=true;
        xajax.$("ButtonFormSelectRegion").value="Подождите...";
        xajax_Result_Select(xajax.getFormValues("FormSelectRegion"));
        enctype="multipart/form-data";>
<input type='hidden' id='number' name='number' value='0'><br>
<input type='hidden' id='vibor' name='vibor' value='0'><br>
<b> Выбор шаблона вывода </b><br>
<!--выбор шаблона-- >
<div>
<select id=template name=template>
  <option value=1>шаблон 1
  <option value=2 selected>шаблон 2
  <option value=3>шаблон 3
</select>
</div>
<b> Выбор региона </b><br>
<div id='divselectregion1'>
<select name=selectregion1 id='selectregion1' >
{foreach from=$info item=item}
```

```
<option value={$item.id} >{$item.name} {$item.socr}
{/foreach}
</select></div>
<input type='submit' id='ButtonFormSelectRegion' value='Выбрать' >
</form>
<div id='div_result'></div>
</body>
</html>
```

Выбираем нужный шаблон, затем нужный регион и нажимаем кнопку **Выбрать**. При этом вызывается хайп-функция `Result_Select`, которой передаются параметры формы. Функция собирает результаты в переменной-массиве `$info`. Далее передаем это значение для переменной Smarty:

```
$smarty->assign("info", $info);
```

И отправляем все это в один из шаблонов Smarty в зависимости от значения, выбранного в форме. Обратите внимание, что обработанный шаблон передается в переменную:

```
$content=$smarty->fetch('template2.html');
```

И только потом контент передается в блок результата хайп-функцией:

```
$objResponse->assign("div_result","innerHTML",$content);
```

Функция `Result_Select` находится в файле `result_select.php` (листинг 2.38).

Листинг 2.38

```
<?php
function Result_Select ($Id)
{
    $objResponse = new xajaxResponse();
    // подключение библиотеки Smarty
    //require_once("Smarty/libs/Smarty.class.php");
    // создание нового класса
    $smarty = new Smarty();
    // путь к директории шаблонов
    $smarty->template_dir = 'my_templates';
    $smarty->compile_dir = 'my_templates_c';
    // подключение к базе данных
    require_once("mybaza.php");
    // получение результата
    $query1="SELECT id,name,socr FROM ".TABLE1." WHERE id_punkt=0
        && id_region='".substr($Id[selectregion1],1,2)."'"
        && id_town=0 && id_rayon>0
        ORDER BY id ASC LIMIT 0, 10";
    $rez1=mysql_query($query1);
    $i=0;
```

```

while($row1=mysql_fetch_assoc($rez1))
{
    $i++;
    $info[$i][id]=$row1[id];
    $info[$i][name]=$row1[name];
}
// создание переменной в Smarty
$smarty->assign("info", $info);
// выбор шаблона и получение результата
// выполнения шаблона в переменную
if($Id[template]==1)
    $content=$smarty->fetch('template1.tpl.html');
elseif($Id[template]==2)
    $content=$smarty->fetch('template2.html');
else
    $content=$smarty->fetch('template3.html');
// выдача результата через ajax
$objResponse->assign("div_result","innerHTML",$content);
$objResponse->assign("ButtonFormSelectRegion","value","Выбрать");
$objResponse->assign("ButtonFormSelectRegion","disabled",false);
return $objResponse;
}
?>

```

Примеры к книге (глава 2 пример 8)
Использование Smarty при динамической подгрузке контента

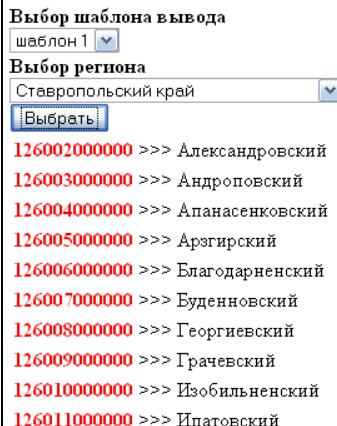


Рис. 2.33. Вывод результата с использованием шаблона template1.tpl.html

Примеры к книге (глава 2 пример 8)
Использование Smarty при динамической подгрузке контента

Выбор шаблона вывода

Выбор региона

*126002000000 --> Александровский
126003000000 --> Андроповский
126004000000 --> Апанасенковский
126005000000 --> Аргирский
126006000000 --> Благодарненский
126007000000 --> Буденновский
126008000000 --> Георгиевский
126009000000 --> Грачевский
126010000000 --> Изобильненский
126011000000 --> Ипатовский*

Рис. 2.34. Вывод результата с использованием шаблона template2.html

Выбор шаблона вывода

Выбор региона

126002000000
Александровский
126003000000
Андроповский
126004000000
Апанасенковский
126005000000
Аргирский
126006000000
Благодарненский
126007000000
Буденновский
126008000000
Георгиевский
126009000000
Грачевский
126010000000
Изобильненский
126011000000
Ипатовский

Рис. 2.35. Вывод результата с использованием шаблона template3.html

На рис. 2.33—2.35 приведен вывод одного и того же результата для разных шаблонов. Файлы шаблонов (template1.tpl.html, template2.html, template3.html)

находятся в папке my_templates. Каталоги для шаблонов и компилированных шаблонов устанавливаются следующим образом:

```
$smarty->template_dir = 'my_templates';
$smarty->compile_dir = 'my_templates_c';
```

Содержимое файлов шаблонов приведено в листинге 2.39.

Листинг 2.39

```
// my_templates/template1.tpl.html
<table>
{foreach from=$info item=item}
<tr style="border-style:;border-width:4">
<td style="color:red;font-weight: bold;">{$item.id}</td>
<td>>> {$item.name}</td>
</tr>
{/foreach}
</table>
// my_templates/template2.html
<table>
{foreach from=$info item=item}
<tr>
<td style="font-style:italic">{$item.id}</td>
<td style="font-style:italic;font-weight: bold;font-size: 20px "> -->
{$item.name}</td>
</tr>
{/foreach}
</table>
// my_templates/template3.html
<table>
{foreach from=$info item=item}
<tr>
<td style="background-color:yellow"><b> {$item.id}</b></td>
</tr>
<tr>
<td>{$item.name}</td>
</tr>
{/foreach}
</table>
```



ЧАСТЬ II

Проект интернет-магазина



Глава 3

Проектирование сайта

3.1. Структура и функции сайта

Сначала определимся с необходимым функционалом создаваемого сайта (интернет-магазина цифровых товаров), рассмотрим особенности разработки сайта без перезагрузки страницы. Создадим разметку главной страницы. Разработаем структуру баз данных создаваемого сайта.

3.1.1. Необходимый функционал сайта (интернет-магазина цифровых товаров)

Прежде всего, нужно продумать, какими возможностями должен обладать современный магазин цифровых товаров. Будем разрабатывать следующий функционал:

□ Блок регистрации:

- "теневая" регистрация для пользователей. Некоторых клиентов от разовой покупки отпугивает процесс регистрации, который длится хотя и не очень долго, но, тем не менее, занимает некоторое время. Не будем терять этих клиентов и предусмотрим возможность покупки, оплаты и автоматического получения товара для незарегистрированных пользователей. Чтобы обеспечить такое, казалось бы невыполнимое, условие, проведем "теневую" регистрацию пользователей. Для этого необходима поддержка cookies в браузере пользователя. "Теневая" регистрация позволит иметь "незарегистрированному" пользователю свой полнофункциональный личный кабинет, в котором будет храниться вся его информация о заказах, покупках и пр. В свой в личный кабинет такой пользователь будет попадать автоматически при заходе на сайт;
- обычная регистрация. Чтобы повысить привлекательность регистрации, введем скидки для зарегистрированных пользователей, добавим при этом оплату отправкой SMS на короткий номер при регистрации, заодно рассмотрим работу сервиса alaggregator.ru.

□ Блок товаров:

- категории товаров неограниченной вложенности;
- просмотр списка товаров по категориям и по фильтру поиска;

- подробный просмотр товаров;
- два типа цен — обычная и цена по акции;
- скидки на товары для различных типов пользователей.

Корзина:

- отбор товаров в корзину по ссылке и перетаскиванием картинки или заголовка товаров (drag&drop);
- корзина должна безразмерной: мало ли что надумает покупатель, может, захочет купить 1000 наименований товара;
- сделаем корзину с "памятью": отложил один товар сегодня, завтра — другой, послезавтра — третий, и все в корзине есть;
- обеспечим подробный вид корзины с возможностью редактирования.

Блок заказов:

- оформление заказа из корзины;
- оплата "электронными" деньгами (Webmoney и ONPAY);
- просмотр всех заказов и их фильтрация;
- редактирование неоплаченных заказов;
- получение оплаченных заказов.

Блок внутренней почты:

- возможность написать и получить сообщения по внутренней почте сайта;
- просмотр сообщений по фильтру.

Мгновенные оповещения на сайте:

- сообщения администратору о новых регистрациях;
- уведомление пользователя о новых сообщениях внутренней почты;
- информация пользователю и администратору о новых заказах;
- сообщения пользователю и администратору о поступивших оплатах по заказам.

Форма обратной связи:

- отправка сообщения на e-mail администратора;
- отправка сообщения на ICQ администратора.

Панель администратора:

- создание, удаление, редактирование категорий товаров;
- создание, удаление, редактирование товаров;
- просмотр всех заказов, редактирование, удаление, изменение статуса заказа;
- редактирование профилей пользователей.

Программирование последнего функционала рассмотрим в *главе 4*.

3.1.2. Структура корневого каталога сайта

Рассмотрим структуру корневого каталога сайта. Организация папок в корневом каталоге сайта приведена в листинге 3.1.

Листинг 3.1

```
/www
    /arhivtovar          # файлы товаров
    /cron                # программы для запуска заданий cron
    /css                 # внешние листы стилей
    /epoch_v106_ru       # класс календаря
    /img                 # картинки оформления сайта
    /imgtovar            # картинки товаров
    /js                  # js файлы
    /prgcontacts         # программы блока Контакты
    /prginfooplata       # программы вывода информации по оплате
    /prgkategory         # формирование категорий
    /prgkorzina          # файлы работы с корзиной
    /prgmessage          # программы сообщений по внутренней почте
    /prgmessage_admin    # программы сообщений по внутренней почте
                         # (админ)
    /prgmessageheader4   # программы блока мгновенных сообщений
    /prgoplata           # программы оплаты заказов
    /prgpopup             # для всплывающих окон
    /prgrate              # выдача курсов валют
    /prgreg               # программы регистрации
    /prgstat              # выдача статистики
    /prgtovars            # программы блока Товары
    /prgtovars_admin     # программы блока Товары (админ)
    /prgusers_admin       # программы блока Пользователи (админ)
    /prgvhod              # программы входа в профиль
    /prgzakaz             # программы блока Заказы
    /prgzakaz_admin      # программы блока Заказы (админ)
    /tmp1                 # хранение файлов корзин пользователей
    /tmparhiv             # временная папка для загрузки файлов
    /tmpimg               # временная папка для загрузки картинок
    /xajax_core            # файлы библиотеки хајах
```

3.1.3. Особенности создания сайта без перезагрузки страницы

Мы создаем сайт, который будет работать без перезагрузки страницы. При открытии сайта загружается index.php — файл главной (и единственной!) страницы.

Страница разделена на блоки (`div`) (рис. 3.1). При переходе по ссылкам перезагрузки страницы не происходит, а идет вызов AJAX-функции по событию `onclick`, например:

```
<a href="javascript:void()" onclick="xajax_Zakaz(1);">Ссылка</a>
```

Далее следует формирование контента на стороне сервера и его подгрузка в соответствующие блоки. Высота блока зависит от его наполняемости и может быть произвольной (листинг 3.2).

Листинг 3.2

```
<?php
$objResponse = new xajaxResponse();
$content="Контент для блока";
$objResponse->assign("center2","innerHTML",$content);
return $objResponse;
?>
```

Если блок нужно сделать невидимым, достаточно для него записать в свойство `innerHTML` данные (HTML код) `<table></table>` (листинг 3.3).

Листинг 3.3

```
<?php
$objResponse = new xajaxResponse();
$content=<table></table>t";
$objResponse->assign("center2","innerHTML",$content);
return $objResponse;
?>
```

Если необходимо выполнить JavaScript-код на стороне клиента, то на стороне сервера выполняем код, приведенный в листинге 3.4.

Листинг 3.4

```
<?php
$objResponse = new xajaxResponse();
// выполнить jQuery-код – фон четных строк таблицы id=table1
// сделать цветом rgb(230,230,230)
$script="$('tr:nth-child(odd)', '#table1').css('background-
color','rgb(230,230,230');");
$objResponse->script($script);
return $objResponse;
?>
```

Кроме рассмотренных, нам потребуются два блока для имитации "всплывающих" окон. Они будут использоваться при написании письма администратору,

перетаскивании методом drag&drop, просмотре пользовательского сообщения, цен на тарифы отправки SMS для разных операторов и пр. Создадим два блока div со свойством position:absolute, z-index:900 и сделаем их первоначально невидимыми (свойство display:none).

header1 (Шапка)		
header2 (Форма для входа)		
header4 (Блок мгновенных сообщений)		
header3 (Пункты главного меню)		
<u>leftcaption1</u>	<u>centercaption1</u>	<u>rightcaption1</u>
left1	center1	right1
<u>leftcaption2</u>	<u>centercaption2</u>	<u>rightcaption2</u>
left2	center2	right2
<u>leftcaption3</u>	<u>centercaption3</u>	<u>rightcaption3</u>
left3	center3	right3
<u>leftcaption4</u>	<u>centercaption4</u>	<u>rightcaption4</u>
left4	center4	right4
<u>leftcaption5</u>	<u>centercaption5</u>	<u>rightcaption5</u>
left5	center5	right5
Bottom (Подвал)		

Рис. 3.1. Разметка страницы index.php

3.1.4. Проектирование базы данных

Для реализации полнофункционального скрипта интернет-магазина создадим следующие таблицы в нашей базе данных:

- users — пользователи;
- mainmenu — распределение пунктов главного меню для пользователей и набор подпрограмм для каждого пункта меню;
- category — категории товаров;
- tovars — товары;
- zakaz — информация о заказе (дата, пользователь, статус, сумма и т. д.);
- zakaz_table — табличная часть заказов;
- link_downloads — таблица ссылок и доступов для скачиваемых товаров;
- messages — сообщения по внутренней "почте";
- message_header4 — мгновенные сообщения на сайте;

- `oplata_reg` — учет SMS при регистрации;
- `oplata` — учет оплаты за товар "электронными" деньгами;
- `link_partners` — ссылки на сайты партнеров;
- `rate` — курсы валют (при парсинге с сайта www.cbr.ru).

Рассмотрим подробно структуру таблиц.

Почти во всех таблицах есть поле `visible` — со значениями `yes` и `no` для включения/отключения (видимость/невидимость) записи.

Основная таблица, определяющая структуру сайта, — `mainmenu`, но, чтобы ее понять, начнем рассмотрение с таблицы `users` (пользователи).

Структура таблицы `users`:

- `id` — первичный ключ;
- `data` — дата и время регистрации;
- `login` — логин пользователя;
- `password` — пароль пользователя;
- `type` — тип пользователя:
 - 1 — незарегистрированный пользователь;
 - 2 — зарегистрированный пользователь;
 - 9 — администратор;
- `memory` — зарезервировано;
- `visible` — статус пользователя:
 - `yes` — активен;
 - `no` — не активирован;
 - `block` — заблокирован;
- `discount` — персональная скидка пользователя (по умолчанию 0);
- `email` — e-mail пользователя;
- `ip` — список IP для пользователя, разделителем служит символ ;
(89.67.234.45;123.34.56.89);
- `vizits` — число визитов пользователя.

Листинг 3.5 содержит дамп для создания таблицы `users`.

Листинг 3.5

```
CREATE TABLE 'magazin'.'users' (
  'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
  'data' datetime NOT NULL ,
  'login' varchar( 32 ) NOT NULL ,
  'password' varchar( 12 ) default NULL ,
  'type' set( '1', '2', '8', '9' ) NOT NULL ,
  'memory' int( 1 ) NOT NULL default '0',
  'visible' set( 'yes', 'no', 'block' ) NOT NULL ,
```

```
'discount' float( 2, 0 ) NOT NULL ,
'email' varchar( 30 ) NOT NULL ,
'ip' tinytext,
'vezits' int( 5 ) default '0',
UNIQUE KEY 'id' ( 'id' )

) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Обратите внимание, что длина поля `login` для незарегистрированных пользователей — 32 символа, а для зарегистрированных — 12 символов. Это связано с тем, что для "незарегистрированных" пользователей происходит "теневая", невидимая пользователю регистрация и в поле `login` записывается уникальное 32-символьное значение.

В системе три типа пользователей — незарегистрированный, зарегистрированный и администратор. Соответственно у каждого свои возможности и выбор подпрограмм. Набор пунктов главного меню и набор подпрограмм для каждого пункта меню для разных типов пользователей берется из таблицы `mainmenu`.

Структура таблицы `mainmenu` обеспечивает формирование главного меню и программ для выбранного пункта:

- `id` — первичный ключ;
- `status` — статус пользователя;
- `name` — название пункта ("Товары", "Заказы" ...);
- `sort` — используется для порядка следования пунктов главного меню;
- `prgprg` — список процедур при выборе пункта меню (разделитель — ;);
- `prgdiv` — список блоков, куда запишется результат выполнения процедур из `prgprg`;
- `visible` — видимость пункта:
 - `yes` — включено;
 - `no` — отключено;
- `img` — картинка для пункта меню.

Листинг 3.6 содержит дамп для создания таблицы `mainmenu`.

Листинг 3.6

```
CREATE TABLE 'magazin'.'mainmenu' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'status' set( '1', '2', '8', '9' ) default NULL ,
'name' varchar( 50 ) default NULL ,
'sort' int( 5 ) NOT NULL ,
'prgdiv' mediumtext,
'prgprg' mediumtext,
'visible' set( 'yes', 'no' ) NOT NULL default 'yes',
'img' varchar( 30 ) default NULL ,
```

```
UNIQUE KEY 'id' ( 'id' ) ,
KEY 'sort' ( 'sort' )

) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

При выборе пункта главного меню вызывается хайх-функция с аргументом, равным значению поля `id` в таблице `mainmenu`. Из базы данных получаем соответствующую строку, выполняем по списку процедуры из поля `prgprg`, а результаты заносятся в блоки из поля `prgdiv`.

Структура таблицы `kategory` — список категорий товаров произвольной вложенности:

- `id` — первичный ключ;
- `id_parent` — ID родительской категории;
- `name` — название категории;
- `nn` — счетчик товаров в категории;
- `visible` — статус видимости:
 - `yes` — включена;
 - `no` — отключена.

Листинг 3.7 содержит дамп для создания таблицы `kategory`.

Листинг 3.7

```
CREATE TABLE 'magazin'.'kategory' (
  'id' int( 5 ) NOT NULL AUTO_INCREMENT ,
  'id_parent' int( 5 ) NOT NULL default '0',
  'name' varchar( 30 ) NOT NULL default '',
  'nn' int( 5 ) NOT NULL default '0',
  'visible' set( 'yes', 'no' ) NOT NULL default 'yes',
  UNIQUE KEY 'id' ( 'id' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Значение поля `id_parent=0` — для корневой папки, `nn` — количество товаров во всех вложенных категориях. Это поле будем изменять программно при добавлении, скрытии, открытии товара.

Структура таблицы `tovars` — информация по товарам:

- `id` — первичный ключ;
- `id_kategory` — ID категории товара;
- `arhiv` — путь к архиву;
- `img` — картинка товара крупная ("превью" будет создаваться динамически при выводе);
- `name` — наименование товара;
- `pay_rub` — цена товара, руб.;
- `new_pay_rub` — цена товара, руб. (для акции);

- data — дата и время занесения товара на сайт;
- data_update — дата и время последнего изменения товара;
- views — количество просмотров товара;
- pays — количество покупок товара;
- rating — рейтинг товара;
- visible — статус видимости:
 - yes — товар "видим" при просмотре списка товаров;
 - no — товар "невидим";
- artikul — артикул (не используется);
- ean13 — код ean13 (не задействован).

Листинг 3.8 содержит дамп для создания таблицы tovars.

Листинг 3.8

```
CREATE TABLE 'magazin'.'tovars' (
'id' int( 5 ) NOT NULL AUTO_INCREMENT ,
'id_kategory' int( 11 ) default '0',
'arhiv' varchar( 50 ) NOT NULL default '',
'img' varchar( 50 ) NOT NULL default '',
'name' varchar( 50 ) NOT NULL default '',
'info' tinytext NOT NULL ,
'fullinfo' text,
'pay_rub' float( 10, 2 ) default '0.00',
'new_pay_rub' float( 10, 2 ) default '0.00',
'data' datetime default '0000-00-00 00:00:00',
'data_update' datetime default '0000-00-00 00:00:00',
'vews' int( 5 ) NOT NULL default '0',
'pays' int( 5 ) default '0',
'rating' float( 8, 2 ) NOT NULL default '0.00',
'vesible' set( 'yes', 'no', 'del' ) default 'no',
'artikul' varchar( 10 ) NOT NULL ,
'ean13' varchar( 13 ) NOT NULL ,
UNIQUE KEY 'id' ( 'id' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Структура таблицы zakaz — информация о заказе (шапка):

- id — первичный ключ;
- data — дата и время заказа;
- id_user — название категории;
- summa_rub — сумма в рублях;
- summa_rub_oplata — сумма оплаченная;

- pay — статус оплаты:
 - yes — оплачен;
 - no — не оплачен;
- visible — статус видимости для отображения:
 - yes — "видим";
 - no — не "видим" (удален).

Листинг 3.9 содержит дамп для создания таблицы zakaz.

Листинг 3.9

```
CREATE TABLE 'magazin'.'zakaz' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'data' datetime NOT NULL ,
'id_user' int( 9 ) NOT NULL ,
'summa_rub' float( 10, 2 ) NOT NULL ,
'summa_rub_oplata' float( 10, 2 ) NOT NULL ,
'pay' set( 'yes', 'no' ) NOT NULL default 'no',
'veisible' set( 'yes', 'no' ) NOT NULL ,
UNIQUE KEY 'id' ( 'id' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Структура таблицы zakaz_table — табличная часть заказов:

- id — первичный ключ;
- id_zakaz — ID заказа;
- id_tovar — ID товара;
- pay_rub — цена товара в рублях;
- kol — количество товара;
- summa_rub — сумма;
- summa_rub_oplata — сумма оплаченная;
- id_link — ссылка на таблицу доступа к файлу цифрового товара;
- count_download — число скачиваний.

Если заказ оплачен не полностью, только часть заказа попадает в оплаченные.

Листинг 3.10 содержит дамп для создания таблицы zakaz_table.

Листинг 3.10

```
CREATE TABLE 'magazin'.'zakaz_table' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'id_zakaz' int( 9 ) NOT NULL ,
'id_tovar' int( 9 ) NOT NULL ,
'pay_rub' float( 10, 2 ) NOT NULL ,
'kol' int( 9 ) NOT NULL ,
```

```
'summa_rub' float( 10, 2 ) NOT NULL ,  
'summa_rub_oplata' float( 10, 2 ) NOT NULL ,  
'id_link' int( 9 ) default NULL ,  
'count_download' int( 2 ) default NULL ,  
UNIQUE KEY 'id' ( 'id' )  
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Каждая ссылка на товар привязана к файлу товара и к пользователю. Доступ к файлу ссылки прописывается для каждого пользователя (для всех IP пользователя). Поле `data` может служить для ограничения доступа к ссылке по времени. Структура таблицы `link_downloads` — таблица ссылок и доступов для скачиваемых товаров:

- `id_link` — первичный ключ;
- `id_user` — ID пользователя;
- `id_zakaz` — ID заказа;
- `file` — путь к файлу;
- `status` — статус:
 - `yes` — скачивание разрешено;
 - `no` — скачивание недоступно;
- `data` — дата и время доступа к скачиванию.

Листинг 3.11 содержит дамп для создания таблицы `link_downloads`.

Листинг 3.11

```
CREATE TABLE 'magazin'.'link_downloads' (  
'id_link' int( 9 ) NOT NULL AUTO_INCREMENT ,  
'id_user' int( 9 ) NOT NULL ,  
'id_zakaz' int( 9 ) NOT NULL ,  
'file' varchar( 60 ) NOT NULL ,  
'status' set( 'yes', 'no' ) NOT NULL ,  
'data' date default NULL ,  
UNIQUE KEY 'id_link' ( 'id_link' )  
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Структура таблицы `messages` — сообщения по внутренней "почте":

- `id` — первичный ключ;
- `data` — дата и время создания сообщения;
- `id_to` — ID пользователя (отправитель);
- `id_from` — ID пользователя (получатель);
- `theme` — тема;
- `message` — сообщение;
- `view_to` — статус:
 - `yes` — прочитано;
 - `no` — не прочитано;

visible_to — статус:

- yes — "видимо" для получателя;
- no — "невидимо" для получателя;

visible_from — статус:

- yes — "видимо" для отправителя;
- no — "невидимо" для отправителя.

Листинг 3.12 содержит дамп для создания таблицы messages.

Листинг 3.12

```
CREATE TABLE 'magazin'.'messages' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'data' datetime NOT NULL ,
'id_to' int( 9 ) NOT NULL ,
'id_from' int( 9 ) NOT NULL ,
'theme' varchar( 30 ) NOT NULL ,
'message' tinytext NOT NULL ,
'vew_to' set( 'yes', 'no' ) NOT NULL default 'no',
'veisible_to' set( 'yes', 'no' ) default 'yes',
'veisible_from' set( 'yes', 'no' ) default 'yes',
UNIQUE KEY 'id' ( 'id' ) ,
KEY 'data' ( 'data' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Мгновенные сообщения на сайте будем создавать при формировании заказа, поступлении оплаты за заказ, при отправке сообщения по внутренней почте сайта, для администратора при регистрации новых пользователей, для пользователей, зашедших первый раз на сайт, с предложением зарегистрироваться. При просмотре сообщения будем устанавливать признак visible=no, и повторно оно появляться не будет. Структура таблицы message_header4 — мгновенные сообщения на сайте:

id — первичный ключ;

data — дата и время создания сообщения;

id_user — ID пользователя, для кого создано сообщение;

type — тип сообщения:

- 1 — ссылка на форму регистрации;
- 2 — ссылка на регистрацию (данные клиента);
- 3 — зарезервировано;
- 4 — ссылка на личное сообщение;
- 5 — ссылка на заказ (созданный заказ);
- 6 — ссылка на заказ (поступившую оплату);

arg_id — ID аргумента ссылки (ID заказа, ID пользователя, ID сообщения);

- message — сообщение;
- visible — статус:
 - yes — для показа;
 - no — прочитано.

Листинг 3.13 содержит дамп для создания таблицы message_header4.

Листинг 3.13

```
CREATE TABLE 'magazin'.'message_header4' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'data' datetime NOT NULL ,
'id_user' int( 9 ) NOT NULL ,
'type' set( '1', '2', '3', '4', '5', '6' ) NOT NULL ,
'arg_id' int( 9 ) NOT NULL ,
'message' tinytext NOT NULL ,
'visible' set( 'yes', 'no' ) NOT NULL ,
UNIQUE KEY 'id' ( 'id' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Структура таблицы oplata_reg — учет SMS при регистрации:

- id — первичный ключ;
 - data — дата и время отправки SMS-сообщения;
 - phone — телефон, с которого отправлено SMS-сообщение;
 - kod_reg — код активации (отправленный на телефон отправителя SMS);
 - activ — для исключения повторной активации по одному коду:
 - yes — использовался код;
 - no — не использовался код;
- id_user — ID пользователя, использовавшего код активации.

Листинг 3.14 содержит дамп для создания таблицы oplata_reg.

Листинг 3.14

```
CREATE TABLE 'magazin'.'oplata_reg' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'data' datetime default NULL ,
'phone' varchar( 12 ) NOT NULL ,
'kod_reg' varchar( 12 ) NOT NULL ,
'activ' set( 'yes', 'no' ) NOT NULL ,
'id_user' int( 9 ) NOT NULL ,
UNIQUE KEY 'id' ( 'id' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

При отправке сообщения на короткий номер, сервис отправляет на наш скрипт данные отправителя — его номер телефона. В таблице будем сохранять пароль для входа, отправляемый на телефон пользователя, а также признак использования кода, для предотвращения многократной регистрации при отправке одного SMS-сообщения.

В таблице `oplata` будем сохранять параметры оплаты заказа, при оплате "электронными" деньгами. Структура таблицы `oplata`:

- `id` — первичный ключ;
- `data` — дата и время оплаты;
- `id_zakaz` — ID заказа, по которому производилась оплата;
- `plat_system` — система, через которую производилась оплата;
- `kod` — внутренний номер заказа в системе оплаты;
- `summa_rub` — сумма оплаты в рублях.

Листинг 3.15 содержит дамп для создания таблицы `oplata`.

Листинг 3.15

```
CREATE TABLE 'magazin'.'oplata' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'data' datetime NOT NULL ,
'id_zakaz' int( 9 ) NOT NULL ,
'plat_system' set( 'wm', 'ya', 'sms' ) NOT NULL ,
'kod' varchar( 32 ) default NULL ,
'summa_rub' float( 10, 2 ) NOT NULL ,
UNIQUE KEY 'id' ( 'id' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Ссылки на другие сайты целесообразно хранить в базе данных, поэтому для ссылок создаем отдельную таблицу. Структура таблицы `link_partners` — ссылки на сайты партнеров:

- `id` — первичный ключ;
- `name` — текст ссылки;
- `link` — URL ссылки;
- `sort` — поле для сортировки ссылок;
- `visible` — статус ссылки:
 - `yes` — включена;
 - `no` — отключена.

Листинг 3.16 содержит дамп для создания таблицы `link_partners`.

Листинг 3.16

```
CREATE TABLE 'magazin'.'link_partners' (
'id' int( 5 ) NOT NULL AUTO_INCREMENT ,
'name' varchar( 50 ) NOT NULL ,
```

```
'link' varchar( 50 ) NOT NULL ,
'sort' int( 3 ) NOT NULL ,
'visible' set( 'yes', 'no' ) NOT NULL default 'yes',
UNIQUE KEY 'id' ( 'id' ) ,
KEY 'sort' ( 'sort' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Чисто технически курсы валют несложно получать с сайта www.cbr.ru при заходе пользователя на наш сайт, затем обрабатывать и тут же выводить. Но для экономии трафика курсы валют будем получать, запуская скрипт утилитой cron один раз в сутки и сохранять в таблице. Структура таблицы `rate` — курсы валют:

- id — первичный ключ;
- data — дата курса;
- usd — курс доллара;
- eur — курс евро.

Листинг 3.17 содержит дамп для создания таблицы `rate`.

Листинг 3.17

```
CREATE TABLE 'magazin'.'rate' (
'id' int( 9 ) NOT NULL AUTO_INCREMENT ,
'data' date NOT NULL ,
'usd' float( 10, 3 ) default NULL ,
'eur' float( 10, 3 ) default NULL ,
UNIQUE KEY 'id' ( 'id' ) ,
KEY 'data' ( 'data' )
) ENGINE = MYISAM AUTO_INCREMENT = 1 DEFAULT CHARSET = cp1251
```

Вся предварительная работа выполнена. Теперь дело за реализацией — программированием пользовательского интерфейса.

3.2. Типы пользователей. Вход в профиль

3.2.1. Типы пользователей

В системе представлены три типа пользователей:

- незарегистрированный пользователь;
- зарегистрированный пользователь;
- администратор.

Возможности пользователей разных типов различны.

Зарегистрированный пользователь входит в личный кабинет с помощью логин-формы и имеет следующие возможности:

- просмотр товаров, отбор товаров в корзину;
- создание заказов, редактирование заказов;

- оплата заказов;
- общение по внутренней почте.

Незарегистрированный пользователь не является просто гостем, с помощью процесса "теневой регистрации" (см. разд. 3.3.2) он получает такие же права и возможности, как зарегистрированный пользователь.

ЗАМЕЧАНИЕ

Для успешного прохождения "теневой регистрации" необходимо включение cookies на компьютере пользователя.

Возможности администратора:

- создание новых товаров, редактирование и скрытие товаров, определение цен, введение акций по товарам;
- создание, редактирование категорий товаров;
- просмотр профилей пользователей, статистики по пользователям;
- работа с заказами всех пользователей;
- общение по внутренней почте.

3.2.2. Вход в профиль

Мы ставим задачу иметь возможность продавать товар в магазине незарегистрированным пользователям. Как уже упоминалось, для этого необходимо включение cookies на компьютере пользователя. При первом заходе на сайт программа проверяет наличие cookies пользователя (`$_COOKIE["session"]`). При отсутствии cookies происходит "теневая" регистрация пользователя, логин получает значение `session_id()`, тип пользователя — "незарегистрированный пользователь" и значения cookies записываются на компьютер пользователя. При повторном входе на сайт по данным cookies пользователь идентифицируется. Это позволяет для незарегистрированного пользователя иметь свой "личный кабинет" с историей заказов, корзиной и пр. При входе на сайт "незарегистрированного пользователя" в блоке мгновенных сообщений ему предлагается пройти регистрацию. Для повышения привлекательности регистрации, установим скидку на товар для зарегистрированных пользователей. При входе на сайт зарегистрированным пользователям необходимо ввести логин и пароль. Затем выполняется вызов функции `ajax_Vhod` с передачей ей данных формы. При получении пустого логина или пароля, сразу выводим сообщение "Пароль и логин не могут быть пустыми". Далее необходимо сделать запрос в базу данных на наличие в базе `users` пользователя с переданными данными (логином и паролем). Если такой пользователь отсутствует, выводим сообщение "Неверный логин или пароль". Далее проверяется статус пользователя (поле `visible`). При `visible=no` (не активирован) или `visible=block` (блокирован) выводится соответствующее сообщение. Если статус пользователя `visible=yes` (активен), нам необходимо установить переменные `SESSION`, сформировать вид главного меню для данного пользователя и для первого пункта меню выполнить загрузку программ в соответствующие блоки. Функция `ajax_Vhod` находится в файле `prgVhod/vhod.php` (листинг 3.18).

Листинг 3.18

```
function Vhod($Id)
{
    $objResponse = new xajaxResponse();
    // установить флаг запрета для следующих аjax-запросов
    $objResponse->assign("flag_ajax","value",'yes');
    // подключиться к базе mySQL
    require_once("mybaza.php");
    //Далее проверяем логин - пароль
    //пустое поле - возврат
    if(trim($Id['login'])=='' || trim($Id['password'])=='')
    {
        $objResponse->alert("Пароль и логин не могут быть пустыми !");
        $objResponse->assign("ButtonFormVhod","value", "Вход ->");
        $objResponse->assign("login","value", "");
        $objResponse->assign("password","value", "");
        $objResponse->assign("ButtonFormVhod","disabled", false);
        return $objResponse;
    }
    // Далее ищем в таблице USERS пользователя с таким login-password
    $query0="SELECT * FROM users WHERE
        password='".utf8towin($Id['password'])."'
        && login='".utf8towin($Id['login'])."' ";
    $rez0=mysql_query($query0);
    $row0=mysql_fetch_assoc($rez0);
    if(mysql_num_rows($rez0)<1)
    {
        $objResponse->alert("Неверный логин или пароль !");
        $objResponse->assign("ButtonFormVhod","value", "Вход ->");
        $objResponse->assign("login","value", "");
        $objResponse->assign("password","value", "");
        $objResponse->assign("ButtonFormVhod","disabled", false);
        return $objResponse;
    }
    // статус = no - профиль не активирован
    if($row0[visible]=='no')
    {
        $objResponse->alert("Профиль не активирован !");
        $objResponse->assign("ButtonFormVhod","value", "Вход ->");
        $objResponse->assign("login","value", "");
        $objResponse->assign("password","value", "");
        $objResponse->assign("ButtonFormVhod","disabled", false);
        return $objResponse;
    }
```

```
// профиль блокирован
if($row0[visible]=='block')
{
    $objResponse->alert("Профиль блокирован - обратитесь к
        администратору !");
    $objResponse->assign("ButtonFormVhod","value", "Вход ->");
    $objResponse->assign("login","value", "");
    $objResponse->assign("password","value", "");
    $objResponse->assign("ButtonFormVhod","disabled", false);
    return $objResponse;
}

// И наконец - удачно !
// Приступаем к формированию страниц для пользователя
// Установим переменные SESSION
// тип пользователя
$_SESSION[type]=$row0[type];
// id пользователя
$_SESSION[user]=$row0[id];
// Для каждого типа пользователя у нас существует свое главное меню.
// Выбирать пункты главного меню будем из таблицы MAINMENU.
// Формирование персонального меню
$query1="SELECT id,name FROM mainmenu WHERE
    status='".$SESSION[type]."'
    && visible='yes' ORDER BY sort ASC ";
$rez1=mysql_query($query1);$i=1;
while($row1=mysql_fetch_assoc($rez1))
{
    if($i==1)
    {
        // активный пункт меню
        $dataheader3="      ".$row1[name];
    }
    else
        // неактивный пункт меню - формирование ссылки
        $dataheader3="    <a href='
            javascript:void(null)' onclick='
                xajax_MainMenu(\".$row1[id].\");'>".$row1[name]."</a>";
    $i++;
}
// запись в блок Главного меню - header3
$objResponse->assign("header3","innerHTML",$dataheader3);
// Каждый пункт главного меню является ссылкой
// <a href='javascript:void(null)' onclick='
//         xajax_MainMenu(\".$row1[id].\");'>".$row1[name]."</a>";
// Затем получаем список программ для первого пункта главного меню и
```

```
// список блоков,
// куда записывать результаты выполнения.
// Заполнение массивов данными открытого пункта меню
$query20="SELECT id FROM mainmenu WHERE status='".$SESSION[type]."'"
        ORDER BY sort ASC LIMIT 1 ";
$rez20=mysql_query($query20);
$query2="SELECT prgdiv,prgprg,id FROM mainmenu
        WHERE id=".mysql_result($rez20,0)." ";
$rez2=mysql_query($query2);
$row2=mysql_fetch_row($rez2);
$arraydiv=explode(";", $row2[0]);
$arrayprg=explode(";", $row2[1]);
for ($i=0;$i<count($arrayprg);$i++)
{
    $divcontent="";
    switch(trim($arrayprg[$i]))
    {
        // очистить блок
        case "clear": $divcontent=<table></table>;
            break;
        case "clearkorzina": $divcontent=<table></table>;
            $script= "document.getElementById(
                'flag_korzina').value='no';";
            break;
        // заголовок "редактирование категорий" (админ)
        case "zagadmingategory": $divcontent=f_zag1 ("Редактирование
            категорий");
            break;
        // заголовок "редактирование категорий" (админ)
        case "adminkategory":
            $divcontent=<div class='menu'
                id='admin_path_kategory'></div>
            <div class='menu' style='margin-left:0'
                id='admin_kategory1'></div>";
            break;
        // открытие категорий (админ)
        case "adminopencategory":
            $divcontent=f_admin_open_kategory(1);
            break;
        // путь категории (админ)
        case "adminpathcategory":
            $divcontent=<b>Текущая категория
            :</b><br>".$f_string_kategory(1);
            $divcontent.= "<br><a href='javascript:void();' onclick='
                xajax_Delete_Admin_Kategory(1);'>Удалить</a>";
            $divcontent.= "<br><a href='javascript:void();' onclick='
                xajax_EditAdminCategory(1);'>Изменить</a>";
            break;
    }
}
```

```

        xajax_Rename_Admin_Kategory(1);'
    Переименовать</a>";
$divcontent."<br><a href='javascript:void();' onclick='
        xajax_Add_Admin_Kategory(1);'>Добавить</a>";
break;

... И далее ...

case "zaginfooplate": $divcontent=f_zag1("Оплата");
break;
case "infooplate": $divcontent1=f_info_oplate();
$divcontent=$divcontent1;
break;
}
// занести данные в соответствующий блок
$objResponse->assign($arraydiv[$i],"innerHTML",$divcontent);
}
// заменить форму входа приветствием
$dataheader2="Добро пожаловать, <b>".$row0[login]."<b>";
$objResponse->assign("header2","innerHTML",$dataheader2);
// установить флаг - разрешения
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}

```

При нажатии на ссылку пункта главного меню происходит выполнение подпрограммы `xajax_MainMenu(N)`, которая выбирает список подпрограмм и блоков для данного пункта меню. Полный текст процедуры содержится на прилагаемом компакт-диске (файл `prgvhod/vhod.php`).

3.2.3. Использование переменных *SESSION* и *cookies*

Сессии и cookies предназначены для хранения сведений о пользователях при переходах между несколькими страницами. При использовании сессий данные сохраняются во временных файлах на сервере. Файлы с cookies хранятся на компьютере пользователя, и по запросу отсылаются браузером серверу. Сессии и cookies очень удобны и оправданы в таких приложениях, как интернет-магазины, форумы, доски объявлений, когда, во-первых, необходимо сохранять информацию о пользователях на протяжении нескольких страниц, а во-вторых, своевременно предоставлять пользователю новую информацию.

HTTP — это протокол "без сохранения состояния", т. е. он не имеет встроенного способа сохранения состояния между двумя транзакциями. Когда пользователь открывает сначала одну страницу сайта, а затем переходит на другую страницу того же сайта, то, основываясь только на средствах, предоставляемых протоколом HTTP, невозможно установить, что оба запроса относятся к одному пользователю. Следо-

вательно, необходим метод, позволяющий отслеживать информацию о пользователе в течение одного сеанса связи с Web-сайтом. Один из таких методов — управление сессиями при помощи предназначенных для этого функций. Для нас важно то, что сеанс, по сути, представляет собой группу переменных, которые, в отличие от обычных переменных, сохраняются и после завершения выполнения PHP-сценария.

3.2.3.1. Переменные session

Этапы при работе с сессиями:

- открытие сессии;
- регистрация переменных сессии и их использование;
- закрытие сессии.

Самый простой способ открытия сессии — вызов функции `session_start` в начале PHP-сценария:

```
session_start();
```

Эта функция проверяет, существует ли идентификатор сессии, и если нет, то создает его. Если идентификатор текущей сессии уже существует, то загружаются зарегистрированные переменные сессии.

После инициализации сессии появляется возможность сохранять информацию в суперглобальном массиве `$_SESSION`. Пусть имеется файл `page1.php`, в котором в массив `$_SESSION` сохраняется переменная (листинг 3.19).

Листинг 3.19

```
<?php
    // Инициируем сессию
    session_start();
    // Помещаем значение в сессию
    $_SESSION[param1] = "123456789";
    // Выводим ссылку на другую страницу
    echo "<a href='page2.php'>другая страница</a>";
?>
```

На страницах, где происходит вызов функции `session_start()`, значения данных переменных можно извлечь из суперглобального массива `$_SESSION`. В листинге 3.20 приведено содержимое страницы `page2.php`, где извлекаются данные, ранее помещенные на странице `page1.php`.

Листинг 3.20

```
<?php
    // Инициируем сессию
    session_start();
    // Выводим содержимое переменной $_SESSION[param1]
    echo "$_SESSION[param1]=".$_SESSION[param1];
?>
```

После завершения работы с сессией сначала нужно разрегистрировать все переменные сессии, а затем вызвать функцию `unset()`:

```
unset($_SESSION[param1]);
```

3.2.3.2. Переменные cookie

Механизм cookies удобен как для программистов, так и для пользователей. Пользователи выигрывают за счет того, что им не приходится каждый раз заново вводить информацию о себе, а программистам cookies помогают легко и надежно сохранять информацию о пользователях. Cookies — это текстовые строки, хранящиеся на стороне клиента и содержащие пары "имя-значение", с которыми связан URL, по которому браузер определяет, нужно ли посыпать cookie на сервер.

Установка cookies производится с помощью функции `setcookie`. Синтаксис:

```
bool setcookie (string name [, string value [, int expire [, string path  
[, string domain [, int secure]]]]])
```

Аргументы функции `setcookie`:

- `name` — имя устанавливаемой переменной cookie;
- `value` — значение, хранящееся в cookie с именем `$name`;
- `expire` — время в секундах с начала эпохи, по истечении которого текущая переменная cookie становится недействительной;
- `path` — путь, по которому доступна cookie;
- `domain` — домен, из которого доступна cookie;
- `secure` — директива, определяющая, доступна ли переменная cookie не по запросу HTTPS. По умолчанию эта директива имеет нулевое значение, что означает возможность доступа к cookie по обычному запросу HTTP.

ВНИМАНИЕ!

При работе с cookies необходимо учитывать важный момент: переменные cookie нужно обязательно устанавливать перед отправкой в браузер каких-либо заголовков, поскольку сами cookies имеют вид заголовков. Если установить cookies после какого-либо текста, отправляемого в браузер, то возникнет ошибочная ситуация.

Значение, хранящееся в переменной cookie, можно получить через глобальный массив `$_COOKIE[]`. Поскольку некоторые пользователи отключают cookie в настройках своих браузеров, для корректной работы, в приложение, задействующее cookies, необходимо помещать код, проверяющий, включены ли cookies у посетителя, и если нет, то сообщающий ему о необходимости включить cookie (листинг 3.21).

Листинг 3.21

```
<?  
if (!$cookie)  
{  
    // посыпаем заголовок переадресации на страницу,  
    // с которой будет предпринята попытка установить cookie
```

```
header("Location: $PHP_SELF?cookie=1");
// устанавливаем cookie с именем "test"
setcookie("test","1");
}
else
{
if(!$test)
{
echo ("Необходимо включить cookies");
}
else
{
// cookie включены, переходим на нужную страницу
header("Location: http://localhost/test1.php");
}
}
?>
```

По умолчанию `cookies` устанавливаются на один сеанс работы с браузером, однако можно задать для них более продолжительный срок существования. Это очень удобное и полезное свойство, поскольку в данном случае пользователю не придется предоставлять свои данные вновь при каждом посещении сайта.

Как уже говорилось, срок годности устанавливается в секундах относительно начала эпохи. В PHP существуют функции `time` и `mktime` для работы с датой и временем, позволяющие переводить текущее время в число секунд с начала эпохи. Функция `time` просто переводит текущее системное время в число секунд, прошедших с начала эпохи. Усовершенствованный вариант — функция `mktime`:

```
int mktime ([int hour [, int minute [, int second [, int month [, int
day [, int year [, int is_dst]]]]]])
```

Аргумент `is_dst` определяет, попадает ли эта дата в период летнего времени и может принимать следующие значения:

- 1 (по умолчанию; означает, что свойство не задано);
- 0 (временной интервал не находится на период летнего времени);
- 1 (временной интервал находится на период летнего времени).

Примеры

```
<?
// этот cookie действителен в течение 20 мин после создания
setcookie("name", $value, time() + 1200);
// действие этого cookie прекращается в полночь 25 января 2011 года
setcookie("name", $value, mktime(0,0,0,01,25,2011));
?>
```

Удалить переменную `cookie` просто — нужно вызвать функцию `setcookie` и передать ей имя той переменной `cookie`, которая подлежит удалению:

```
setcookie("param1");
```

Другие установленные `cookie` при этом не удаляются.

Иногда в `cookies` приходится хранить конфиденциальные данные, и в этом случае разработчик должен позаботиться о том, чтобы информация, хранящаяся в `cookie`, не была передана третьим лицам. Существует несколько методов защиты информации, хранящейся в `cookie`:

- установка области видимости `cookies`;
- шифрование;
- ограничение доступа для доменов;
- отправка `cookies` по защищенному запросу.

Наилучшее решение — комплексное применение всех этих способов.

Поскольку, по умолчанию, доступ к `cookie` происходит из корневого каталога, это может создать "дыры" в системе защиты, т. к. `cookies` становятся доступными в любом подкаталоге этого каталога. Ограничить доступ к `cookies` для всех страниц, кроме расположенных в конкретном каталоге, например `/catalog`, можно следующим образом:

```
setcookie("name", $value, "/catalog/");
```

Но при этом каталоги `/catalog/index.php`, `/catalog/page.html` и т. д. тоже будут удовлетворять введенному ограничению. Если такое положение нежелательно, можно ограничить область видимости `cookies` до конкретной страницы:

```
setcookie("name", $value, "/catalog/index.php");
```

Для дополнительной безопасности, список доменов, имеющих доступ к `cookies`, должен быть ограничен:

```
setcookie("param1", $value, "/catalog/index.php", ".site.ru");
```

При таком ограничении заданной области видимости будут соответствовать домены с именами `site.ru`, `mysite.ru`, поскольку проверка на допустимость области видимости домена осуществляется по принципу концевого соответствия.

Для переменной `cookie`, хранящей секретные данные, желательно разрешить отвечать только на защищенные запросы HTTP, т. к. в этом случае значительно затрудняется перехват данных, которыми обмениваются клиент и сервер. Для обеспечения защищенного соединения, функции `setcookie` передается шестой параметр со значением, равным единице:

```
setcookie("param1", $value, time() + 600, "/catalog/", ".site.ru", 1);
```

3.2.4. Логика вызова программ при выборе пункта меню

Для каждого типа пользователей определен свой набор пунктов главного меню. Посмотрим, как реализована логика выбора программ для каждого пункта главного меню. При авторизации пользователя на сайте в переменных `SESSION` сохраняется

значение типа пользователя, которое берется из базы данных (таблица `users` поле `type`). При выборе пункта меню происходит вызов хайп-функции `Mainmenu` с аргументом, значение которого равно `id` записи для пары "тип пользователя — пункт главного меню" в таблице `mainmenu` базы данных. Содержимое файла `mainmenu.php` приведено в листинге 3.22. Из таблицы `mainmenu` получаем значения полей `prgprg` и `prgdiv`. Значение поля `prgprg` — списки наборов подпрограмм, разделенные символом `;`. Значение поля `prgdiv` — списки блоков, куда выводятся результаты наборов подпрограмм, разделенные символом `;`. Преобразуем строки значения в массив и последовательно выбираем из массива подпрограмму, выполняя ее, а результат выводим в соответствующий блок. Если для подпрограммы необходимо выполнить сценарий JavaScript на стороне клиента, то отправляем код на исполнение.

Листинг 3.22

```
function MainMenu($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // формирование меню для пользователя
    $query1="SELECT id,name FROM mainmenu WHERE
        status='".$SESSION[type]."' && visible='yes'
        ORDER BY sort ASC ";
    $rez1=mysql_query($query1);
    while($row1=mysql_fetch_assoc($rez1))
    {
        // активный пункт меню
        if($row1[id]==$Id)
        {
            $dataheader3.="     ".$row1[name];
        }
        else
        // неактивный пункт меню – ссылка
        $dataheader3.="   <a href='javascript:void(null)'
            onclick='
                xajax_MainMenu(\".$row1[id].\");'>".$row1[name]."</a>";
    }
    // вывод пунктов главного меню
    $objResponse->assign("header3","innerHTML",$dataheader3);
    // получить для открытого пункта меню набор программ и блоков
    $query2="SELECT prgdiv,prgprg,id FROM mainmenu WHERE id=\"$Id\" ";
    $rez2=mysql_query($query2);
    $row2=mysql_fetch_row($rez2);
```

```
// перевести строку в массив
$arraydiv=explode(";", $row2[0]);
$arrayprg=explode(";", $row2[1]);
// выполнение программ
for($i=0;$i<count($arrayprg);$i++)
{$divcontent="";
$script="";
switch(trim($arrayprg[$i]))
{
// выход из профиля
case "out":
$_SESSION[type]=1;
$query11="SELECT id,name FROM mainmenu WHERE status='1'
&& visible='yes' ORDER BY sort ASC ";
$rez11=mysql_query($query11);$j=0;
while($row11=mysql_fetch_assoc($rez11))
{$j++;
if($j==1)
$divcontent="       ".$row11[name];
else
$divcontent.="   <a href='
javascript:void()' onclick='xajax_MainMenu(\".$row11[id].\")
;'>".$row11[name]."</a>";
}
$query12="SELECT id FROM users
WHERE login='".$SESSION[session]."' ";
$_SESSION[user]=mysql_result(mysql_query($query12),0);
$script="document.getElementById('flag_time1').
value='".$SESSION[user]."';";
$script.="document.getElementById('header4').innerHTML=
'Нет сообщений';";
break;
case "formvhod": $divcontent=<form name='FormVhod' id='FormVhod'
action='javascript:void(null);' onsubmit='
xajax.\\"ButtonFormVhod\\").disabled=true;
xajax.\\"ButtonFormVhod\\").value=\"Подождите...\\";
xajax_Vhod(xajax.getFormValues(\"FormVhod\"));'>
Логин <input type='text' id='login' name='login' value=''
size=10 maxlength=10>
Пароль <input type='password' id='password'
name='password' value=' size=10 maxlength=10>
<input type='submit' id='ButtonFormVhod' value='Вход ->'>
</form>";
break;
// редактирование категорий (админ)
case "clear": $divcontent=<table></table>";
```

```
        break;
    case "clearkorzina": $divcontent=<table></table>;
        $script="document.getElementById('flag_korzina').value='no';";
        break;
    // редактирование категорий (админ)
    case "zagadminkategory":$divcontent=f_zag1(
        "Редактирование категорий");
        break;
    case "admincategory":
        $divcontent=<div class='menu' id='admin_path_category' >
        </div><div class='menu' style='margin-left:0'
        id='admin_category1'>*****</div>";
        break;
    case "adminopencategory":
        $divcontent=f_admin_open_category(1);
        break;
    case "adminpathcategory":
        $divcontent=<b>Текущая категория :</b><br>
        ".f_string_category(1);
        $divcontent.= "<br><a href='javascript:void();' onclick='
            xajax_Delete_Admin_Kategory(1);'>Удалить</a>";
        $divcontent.= "<br><a href='javascript:void();' onclick='
            xajax_Rename_Admin_Kategory(1);'>Переименовать</a>";
        $divcontent.= "<br><a href='javascript:void();' onclick='
            xajax_Add_Admin_Kategory(1);'>Добавить</a>";
        break;
    // регистрация
    case "zagreg": $divcontent=f_zag1("Регистрация");
        break;
    case "reg": $divcontent=f_form_reg_user();
        break;
    // курсы валют
    case "zagrate": $divcontent=f_zag1("Курсы валют");
        break;
    case "rate": $divcontent=f_view_rate();
        break;
    // категории товаров
    case "zagcategory": $divcontent=f_zag1("Заголовок");
        break;
    case "category":
        $divcontent=<div class='menu' style='margin-left:0'
        id=kategory1></div>";
        break;
    case "opencategory":
        $divcontent=f_open_category(1);
```

```
        break;
    // korzina
    case "zagkorzina": $divcontent=f_zag1("Корзина");
        break;
    case "korzina": $divcontent=f_korzina_right();
        break;
    // alltovars
    case "zagalltovars": $divcontent=f_zag1("Все товары");
        break;
    case "alltovars": $divcontent1=f_view_all_tovars(1);
        $divcontent=$divcontent1[0];
        break;
    case "zagalltovarspage": $divcontent="";
        break;
    case "alltovarspage": $divcontent1=f_view_all_tovars(1);
        $divcontent=$divcontent1[1];
        break;
    // newtovars
    case "zagnewtovars": $divcontent=f_zag1("NEW товары");
        break;
    case "newtovars": $divcontent1=f_view_new_tovars(1);
        $divcontent=$divcontent1[0];
        break;
    case "zagnewtovarspage": $divcontent="";
        break;
    case "newtovarspage": $divcontent1=f_view_new_tovars(1);
        $divcontent=$divcontent1[1];
        break;
    // tovars АКЦИЯ
    case "zagtovarsaction": $divcontent=f_zag1("АКЦИЯ !!!");
        break;
    case "tovarsaction": $divcontent1=f_view_tovars_action();
        $divcontent=$divcontent1;
        break;
    // search
    case "zagsearchtovars": $divcontent=f_zag1("Поиск товаров");
        break;
    case "searchtovars": $divcontent1=f_form_search_tovars();
        $divcontent=$divcontent1;
        break;
    // search zakaz
    case "zagsearchzakaz": $divcontent=f_zag1("Поиск заказов");
        break;
    case "searchzakaz": $divcontent=f_form_search_zakaz();
        $script="calendar1=new Epoch('epoch_popup', 'popup',
```

```
document.getElementById('datazakaz1'));" ;
$script.="calendar2=new Epoch('epoch_popup','popup',
document.getElementById('datazakaz2'));" ;
break;
// viewallzakaz
case "zagallzakaz": $divcontent=f_zag1("Все заказы");
break;
case "allzakaz": $divcontent1=f_view_all_zakaz(1);
$divcontent=$divcontent1[0];
break;
case "zagallzakazpage": $divcontent="";
break;
case "allzakazpage": $divcontent1=f_view_all_zakaz(1);
$divcontent=$divcontent1[1];
break;
// search zakaz_admin
case "zagsearchzakaz_admin": $divcontent=f_zag1("Поиск
заказов (админ)");
break;
case "searchzakaz_admin": $divcontent=f_form_search_zakaz_admin();
$script="calendar1=new Epoch('epoch_popup','popup',
document.getElementById('datazakaz1'));" ;
$script.="calendar2=new Epoch('epoch_popup','popup',
document.getElementById('datazakaz2'));" ;
break;
// viewallzakaz_admin
case "zagallzakaz_admin": $divcontent=f_zag1("Все заказы");
break;
case "allzakaz_admin": $divcontent1=f_view_all_zakaz_admin(1);
$divcontent=$divcontent1[0];
break;
case "zagallzakazpage_admin": $divcontent="";
break;
case "allzakazpage_admin": $divcontent1=f_view_all_zakaz_admin(1);
$divcontent=$divcontent1[1];
break;
// messages + messages_admin
case "zagviewallmessages": $divcontent=f_zag1("Входящие
сообщения");
break;
case "viewallmessages": $divcontent1=f_view_all_messages_in(1);
$divcontent=$divcontent1[0];
break;
case "zagviewallmessagespage": $divcontent="";
break;
```

```
case "viewallmessagespage": $divcontent1=f_view_all_messages_in(1);
    $divcontent=$divcontent1[1];
    break;
case "zagsearchmessage": $divcontent=f_zag1("Поиск сообщения");
    break;
case "searchmessage": $divcontent=f_form_search_message_admin();
    $script="calendar1=new Epoch('epoch_popup','popup',
    document.getElementById('datazakaz1'));";
    $script.="calendar2=new Epoch('epoch_popup','popup',
    document.getElementById('datazakaz2'));";
    break;
// users
case "zagallusers": $divcontent=f_zag1("Все пользователи");
    break;
case "allusers": $divcontent1=f_view_all_users(1);
    $divcontent=$divcontent1[0];
    break;
case "zagalluserspage": $divcontent="";
    break;
case "alluserspage": $divcontent1=f_view_all_users(1);
    $divcontent=$divcontent1[1];
    break;
// search users
case "zagsearchusers": $divcontent=f_zag1("Поиск пользователей");
    break;
case "searchusers": $divcontent1=f_form_search_users();
    $divcontent=$divcontent1;
    break;
// contacts
case "zagcontacts": $divcontent=f_zag1("Контакты");
    break;
case "contacts": $divcontent1=f_view_contacts();
    $divcontent=$divcontent1;
    break;
// tovarspartners
case "zagtovarspartners": $divcontent=f_zag1("У партнеров");
    break;
case "tovarspartners": $divcontent1=f_tovars_partners();
    $divcontent=$divcontent1;
    break;
// stat
case "zagstat1": $divcontent=f_zag1("Статистика");
    break;
case "stat1": $divcontent1=f_stat_site1();
    $divcontent=$divcontent1;
```

```

        break;
    // info oplata
    case "zaginfooplata": $divcontent=f_zag1("Оплата");
        break;
    case "infooplata": $divcontent1=f_info_oplata();
        $divcontent=$divcontent1;
        break;
    }
    // вывести результат в соответствующий блок
    $objResponse->assign($arraydiv[$i], "innerHTML", $divcontent);
    // если есть javascript - отправить на выполнение
    if(strlen($script)>0)
        $objResponse->script($script);
    }
    $objResponse->assign("flag_ajax","value", 'no');
    return $objResponse;
}
?>

```

3.2.5. Набор подпрограмм модулей для разных пользователей

Как мы уже говорили, для каждого типа пользователей определен свой набор пунктов главного меню. И для каждого пункта главного меню задан свой выбор подпрограмм, которые нужно выполнить для формирования контента выбранного пункта меню. Набор программ для каждого пункта меню для разных типов пользователей берем из таблицы `mainmenu` базы данных (поле `prgprg`). Набор блоков вывода контента для каждого набора программ — из поля `prgdiv`. Для наглядного представления составим таблицы выбора подпрограмм при выборе пункта главного меню для разных типов пользователей (табл. 3.1 – 3.3).

Таблица 3.1. Набор программ для незарегистрированного пользователя

Пункт главного меню	Незарегистрированный пользователь		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Товары	Заголовок для категорий	zagcategory	leftcaption1
	Категории товаров	kategory	left1
	Раскрыть категорию	openkategory	kategory1
	Заголовок "Акции"	zagtovarsaction	leftcaption2
	Список товаров акции	tovarsaction	left2
	Заголовок "Курсы валют"	zgrate	rightcaption1

Таблица 3.1 (продолжение)

Пункт главного меню	Незарегистрированный пользователь		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Товары	Курсы валют	rate	right1
	Заголовок "Корзина"	zagkorzina	rightcaption2
	Корзина краткая	korzina	right2
	Заголовок "Товары"	zagalltovars	centercaption3
	Список товаров	alltovars	center3
	Заголовок навигатор страниц для товаров	zagalltovarspage	centercaption4
	Навигатор страниц для товаров	alltovarspage	center4
	Заголовок "Поиск товара"	zagsearchtovars	centercaption2
	Форма поиска товаров	searchtovars	center2
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
	Заголовок "Контакты"	zagcontacts	leftcaption4 rightcaption4
	Контакты	contacts	left4, right4
	Заголовок "Партнеры"	zagtovarspartners	rightcaption3
Заказы	Ссылки партнеров	tovarspartners	right3
	Заголовок "Статистика"	zagstat1	leftcaption3
	Статистика	stat1	left3
	Заголовок "Поиск заказов"	zagsearchzakaz	centercaption2
	Форма поиска заказов	searchzakaz	center2
	Заголовок "Заказы пользователя"	zagallzakaz	centercaption3
	Заказы пользователя	allzakaz	center3
	Заголовок "Навигатор страниц для заказов"	zagallzakazpage	centercaption4
	Навигатор страниц для заказов	allzakazpage	center4

Таблица 3.1 (окончание)

Пункт главного меню	Незарегистрированный пользователь		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Заказы	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Переписка	Заголовок "Сообщения"	zagviewallmessages	centercaption3
	Сообщения пользователя	viewallmessages	center3
	Заголовок "Навигатор страниц для сообщений"	zagviewallmessagespage	centercaption4
	Навигатор страниц для сообщений пользователя	viewallmessagespage	center4
	Очистка блока	clear	centercaption2, center2, centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Оплата	Заголовок "Оплата"	zaginfooplata	centercaption2
	Информация по оплате	infooplata	center2
	Очистка блока	clear	centercaption3, center3, centercaption4, center4, centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Регистрация	Заголовок "Регистрации"	zagreg	centercaption2
	Форма регистрации	reg	center2
	Очистка блока	clear	centercaption3, center3, centercaption4, center4, centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1

Таблица 3.2. Набор программ для зарегистрированного пользователя

Пункт главного меню	Зарегистрированный пользователь		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Товары	Заголовок для категорий	zagcategory	leftcaption1
	Категории товаров	category	left1
	Раскрыть категорию	opencategory	kategory1
	Заголовок "Акции"	zagtovarsaction	leftcaption2
	Список товаров акции	tovarsaction	left2
	Заголовок "Курсы валют"	zagrave	rightcaption1
	Курсы валют	rate	right1
	Заголовок "Корзина"	zagkorzina	rightcaption2
	Корзина краткая	korzina	right2
	Заголовок "Товары"	zagalltovars	centercaption3
	Список товаров	alltovars	center3
	Заголовок навигатор страниц для товаров	zagalltovarspage	centercaption4
	Навигатор страниц для товаров	alltovarspage	center4
	Заголовок "Поиск товара"	zagsearchtovars	centercaption2
	Форма поиска товаров	searchtovars	center2
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
	Заголовок "Контакты"	zagcontacts	leftcaption4 rightcaption4
	Контакты	contacts	left4, right4
Заказы	Заголовок "Партнеры"	zagtovarspartners	rightcaption3
	Ссылки партнеров	tovarspartners	right3
	Заголовок "Статистика"	zagstat1	leftcaption3
	Статистика	stat1	left3
Заказы	Заголовок "Поиск зака- зов"	zagsearchzakaz	centercaption2
	Форма поиска заказов	searchzakaz	center2

Таблица 3.2 (продолжение)

Пункт главного меню	Зарегистрированный пользователь		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Заказы	Заголовок "Заказы пользователя"	Zagallzakaz	centercaption3
	Заказы пользователя	allzakaz	center3
	Заголовок "Навигатор страниц для заказов"	zagallzakazpage	centercaption4
	Навигатор страниц для заказов	allzakazpage	center4
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Переписка	Заголовок "Сообщения"	zagviewallmessages	centercaption3
	Сообщения пользователя	viewallmessages	center3
	Заголовок "Навигатор страниц для сообщений"	zagviewallmessages-page	centercaption4
	Навигатор страниц для сообщений пользователя	viewallmessagespage	center4
	Очистка блока	clear	centercaption2, center2, centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Оплата	Заголовок "Оплата"	zaginfooplata	centercaption2
	Информация по оплате	infooplata	center2
	Очистка блока	clear	centercaption3, center3, centercaption4, center4, centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Выход	Выход из профиля	out	header3
	Форма входа	formvhod	header2
	Заголовок для категорий	zagcategory	leftcaption1

Таблица 3.2 (окончание)

Пункт главного меню	Зарегистрированный пользователь		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Выход	Категории товаров	Category	left1
	Раскрыть категорию	opencategory	category1
	Заголовок "Акции"	zagtovarsaction	leftcaption2
	Список товаров акции	tovarsaction	left2
	Заголовок "Курсы валют"	zagrate	rightcaption1
	Курсы валют	rate	right1
	Заголовок "Корзина"	zagkorzina	rightcaption2
	Корзина краткая	korzina	right2
	Заголовок "Товары"	zagalltovars	centercaption3
	Список товаров	alltovars	center3
	Заголовок "Навигатор страниц для товаров"	zagalltovarspage	centercaption4
	Навигатор страниц для товаров	alltovarspage	center4
	Заголовок "Поиск товара"	zagsearchtovars	centercaption2
	Форма поиска товаров	searchtovars	center2
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1

Набор программ для администратора обширнее. Это обусловлено наличием у администратора прав, отсутствующих у обычных пользователей: редактирование категорий товаров, создание товаров и работа с профилями пользователей.

Таблица 3.3. Набор программ для администратора

Пункт главного меню	Администратор		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Товары	Заголовок для категорий	zagcategory	leftcaption1
	Категории товаров	category	left1
	Раскрыть категорию	opencategory	category1

Таблица 3.3 (продолжение)

Пункт главного меню	Администратор		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Товары	Заголовок "Акции"	Zagtovarsaction	leftcaption2
	Список товаров акции	tovarsaction	left2
	Заголовок "Корзина"	zagkorzina	rightcaption2
	Корзина краткая	korzina	right2
	Заголовок "Новые товары"	zagnewtovars	centercaption3
	Список товаров	newtovars	center3
	Заголовок "Навигатор страниц для новых товаров"	zagnewtovarspage	centercaption4
	Навигатор страниц для новых товаров	newtovarspage	center4
	Заголовок "Поиск товара"	zagsearchtovars	centercaption2
	Форма поиска товаров	searchtovars	center2
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Заказы	Заголовок "Поиск заказов"	zagsearchzakaz_admin	centercaption2
	Форма поиска заказов	searchzakaz_admin	center2
	Заголовок "Заказы всех пользователей"	zagallzakaz_admin	centercaption3
	Заказы всех пользователей	allzakaz_admin	center3
	Заголовок "Навигатор страниц для заказов"	zagallzakaz-page_admin	centercaption4
	Навигатор страниц для заказов	allzakazpage_admin	center4
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1

Таблица 3.3 (продолжение)

Пункт главного меню	Администратор		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Переписка	Заголовок "Сообщения"	zagviewallmessages	centercaption3
	Сообщения пользователей	viewallmessages	center3
	Заголовок "Навигатор страниц для сообщений"	zagviewallmessages-page	centercaption4
	Навигатор страниц для сообщений пользователей	viewallmessagespage	center4
	Заголовок поиска сообщений	zagsearchmessage	centercaption2
	Форма поиска сообщений	searchmessage	center2
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Категории	Заголовок "Редактирование категорий"	zagadminkcategory	centercaption3
	Дерево категорий	adminkcategory	center3
	Открыть дерево	adminopencategory	admin_category1
	Путь к текущей категории	adminpathcategory	admin_path_category
	Очистка блока	clear	centercaption2, center2, centercaption4, center4, centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Пользователи	Заголовок "Пользователи"	zagallusers	centercaption3
	Список пользователей	allusers	center3
	Заголовок "Навигатор страниц для пользователей"	zagalluserspage	centercaption4
	Навигатор страниц для пользователей	alluserspage	center4

Таблица 3.3 (окончание)

Пункт главного меню	Администратор		
	Описание	Набор подпрограмм/ (файлы)	Блок вывода
Пользователи	Заголовок "Поиск пользователей"	zagsearchusers	centercaption2
	Поиск пользователей	searchusers	center2
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1
Выход	Выход из профиля	out	header3
	Форма входа	formvhod	header2
	Заголовок для категорий	zagcategory	leftcaption1
	Категории товаров	kategory	left1
	Раскрыть категорию	openkategory	kategory1
	Заголовок "Акции"	zagtovarsaction	leftcaption2
	Список товаров акции	tovarsaction	left2
	Заголовок "Курсы валют"	zagrate	rightcaption1
	Курсы валют	rate	right1
	Заголовок "Корзина"	zagkorzina	rightcaption2
	Корзина краткая	korzina	right2
	Заголовок "Товары"	zagalltovars	centercaption3
	Список товаров	alltovars	center3
	Заголовок "Навигатор страниц для товаров"	zagalltovarspage	centercaption4
	Навигатор страниц для товаров	alltovarspage	center4
	Заголовок "Поиск товара"	zagsearchtovars	centercaption2
	Форма поиска товаров	searchtovars	center2
	Очистка блока	clear	centercaption5, center5, centercaption1
	Убрать показ корзины подробно	clearkorzina	center1

Далее при работе вызов функций осуществляется по технологии AJAX — по событию `onclick` ссылки вызывается хайп-функция, которая записывает результат в соответствующий блок.

3.3. Регистрация

3.3.1. "Теневая" регистрация незарегистрированных пользователей

Как говорилось ранее, мы ставим задачу иметь возможность продавать товар в магазине незарегистрированным пользователям. Для этого будем проводить "теневую" регистрацию. Необходимо включение `cookies` на компьютере пользователя. При первом заходе на сайт программа проверяет наличие `cookies` пользователя (`$_COOKIE["session"]`). При отсутствии происходит "теневая" регистрация пользователя с логином, равным значению `session_id()`, устанавливается тип пользователя — "незарегистрированный пользователь" и значения `cookies` записываются на компьютер пользователя. При повторном входе на сайт, по данным `cookies` пользователь идентифицируется. Процесс "теневой" регистрации позволит иметь "незарегистрированному" пользователю свой полнофункциональный личный кабинет, в котором будет храниться вся его информация о заказах, покупках и т. п. В свой в личный кабинет такой пользователь будет попадать автоматически при заходе на сайт. Программную реализацию "теневой" регистрации (файл `index.php`) иллюстрирует листинг 3.23.

Листинг 3.23

```
// Инициализируем сессию
session_start();
// соединение с базой данных
require_once("mybaza.php");
// есть куки
if(isset($_COOKIE["session"]))
{
    $_SESSION[session]=$_COOKIE["session"];
    $query1="SELECT id FROM users WHERE login='".$SESSION[session]."' ";
    $rez1=mysql_query($query1);
    if(mysql_num_rows($rez1)>0)
    {
        // получаем параметры пользователя и записываем в
        // переменные SESSION
        $_SESSION[user]=mysql_result($rez1,0);
        $_SESSION[type]=1;
    }
}
else
{
```

```
// создать нового "теневого" пользователя и записать в базу
$data=date('Y-m-d H:i:s');
$query2="INSERT INTO users SET data='".$data."',
    login='".$SESSION[session]."',
    password='',type='1',visible='yes',ip='".$f_get_ip().";' ";
mysql_query($query2);
$_SESSION[user]=mysql_insert_id();
$_SESSION[type]=1;
// сообщение в блок мгновенных сообщений с предложением
// пройти регистрацию
f_create_message_header4(1,0, $_SESSION[user],
"Зарегистрируйтесь и получайте скидку 50 % на все товары ");
// сообщение в блок мгновенных сообщений для администратора
// о "теневой" регистрации нового пользователя
f_create_message_header4(2,$_SESSION[user],0,
"Зарегистрирован гостевой пользоваталь с ip='".$SERVER[REMOTE_ADDR]);
}

}
else
{
    // создать нового "теневого" пользователя и записать в базу
$_SESSION[session]=session_id();
$data=date('Y-m-d H:i:s');
$query1="INSERT INTO users SET da-
ta='".$data."',login='".$SESSION[session]."',
    password='',type='1',visible='yes',ip='".$f_get_ip().";' ";
mysql_query($query1);
$_SESSION[user]=mysql_insert_id();
$_SESSION[type]=1;
// сообщение в блок мгновенных сообщений с предложением
// пройти регистрацию
f_create_message_header4(1,0, $_SESSION[user],"Зарегистрируйтесь и получайте
скидку 50 % на все товары ");
// сообщение в блок мгновенных сообщений для администратора
// о "теневой" регистрации нового пользователя
f_create_message_header4(2,$SESSION[user],0,"Зарегистрирован гостевой поль-
зователь с ip='".$SERVER[REMOTE_ADDR]);
}
// установить cookies на год
setcookie("session",$_SESSION[session],strtotime("now+365 days"));
// счетчик количества визитов пользователя
$query01="SELECT vizits FROM users WHERE id='".$SESSION[user]."' ";
$rez01=mysql_query($query01);
$query02="UPDATE users SET vizits=').(mysql_result($rez01,0)+1).'' WHERE
id='".$SESSION[user]."' ";
mysql_query($query02);
$_SESSION[ip]=$_SERVER['REMOTE_ADDR'];
```

3.3.2. Регистрация пользователей

Для регистрации необходимо нажать на ссылку **Регистрация** основного меню либо ссылку **Зарегистрируйтесь и получайте скидку на все товары** в блоке мгновенных сообщений. При нажатии на ссылку открывается форма регистрации (рис. 3.2).

Рис. 3.2. Форма регистрации пользователя

Файл prgereg/function_form_reg_user.php (листинг 3.24) формирует форму регистрации. Для зарегистрированных клиентов в магазине действует скидка: в файле my.php предусмотрена константа define(DISCOUNT, 30). Сделаем регистрацию платной. Оплата осуществляется посылкой SMS на короткий номер.

ЗАМЕЧАНИЕ

Стоимость отправки SMS для различных операторов можно узнать по ссылке **Стоимость SMS**. При этом вызывается хайп-функция Popup_Tarif_SMS1, расположенная в файле prgorpopup/popup_tarif_sms1.php, которая выводит результат во всплывающее окно. Функция Popup_Tarif_SMS1 приведена в листинге 3.24.

Кнопка **Зарегистрироваться** неактивна до момента правильного заполнения всех полей формы регистрации и введения кода SMS при отправке сообщения на короткий номер. Проверка полей заполняемой формы будет производиться "на лету" средствами AJAX с выдачей сообщения рядом с заполняемым полем (красный цвет — ошибка, синий — верное заполнение).

Листинг 3.24

```
// Вывод формы регистрации пользователя
function f_form_reg_user()
{
    require_once("my.php");
    $text1=<center>;
// ссылка на пользовательское соглашение -
// ajax-функция хаях_Popup_About_Reg (prgpopup/popup_about_reg.php)
$text1.=<br><a href='javascript:void();' onclick='
    хаях_Popup_About_Reg();'>Зарегистрируйтесь и покупайте
    все товары со скидкой ".DISCOUNT." % (регистрация
    платная)</a><br><br>";
$text1.=<form id='FormRegUser' action='javascript:void(null);'
    onsubmit='xajax.\$("ButtonFormRegUser").disabled=true;
    xajax.\$("ButtonFormRegUser").value=\"Подождите...\";
    xajax_Reg_User(xajax.getFormValues(\"FormRegUser\"));'>";
$text1.=<table width=100%>;
// Форма регистрации
$text1.=<caption>Форма регистрации</caption>;
// Поле логин
// Рядом с каждым полем вводим блок div для отображения
// правильности заполнения поля
// по событию onchange
// отправляются все данные формы FormRegUser
$text1.=<tr><td width=50%>Логин (6-12 буквы, цифры)</td>
    <td width=30%><input type='text' name='login' id='login'
        size='20' maxlength='12' value=' ' onchange='
        xajax_Control_Reg_User(xajax.getFormValues(
        \"FormRegUser\"));'></td>
    <td width=20%>
        <div id='reg_user_login'><font color='red'>no</font></div>
    </td></tr>;
// Поле пароль
$text1.=<tr><td width=50%>Пароль (6-12)</td>
    <td width=30%><input type='password' name='password'
        id='password' size='20' maxlength='12' value=' ' onchange='
        xajax_Control_Reg_User(xajax.getFormValues(
        \"FormRegUser\"));'></td>
    <td width=20%>
        <div id='reg_user_password'><font color='red'>
            no</font></div>
    </td></tr>;
// Поле подтверждения пароля
$text1.=<tr><td width=50%>Повторите пароль</td>
```

```
<td width=30%><input type='password' name='password1'
id='password1' size='20' maxlength='12' value='' onchange='
xajax_Control_Reg_User(xajax.getFormValues(
\"FormRegUser\"));'></td>
<td width=20%>
    <div id='reg_user_password1'><font color='red'>
        no</font></div>
    </td></tr>";
// Поле e-mail
$text1.=<tr>
    <td width=50%>
        Ваш e-mail <br>
    </td>
    <td width=30%><input type='input' name='email' id='email'
size='20' maxlength='30' value='' onchange='
xajax_Control_Reg_User(xajax.getFormValues(
\"FormRegUser\"));'></td>
    <td width=20%>
        <div id='reg_user_email'><font color='red'>no</font></div>
    </td></tr>";
// Код, приходящий при отправке SMS-сообщения
$text1.=<tr>
    <td width=50%>
        Код активации <br> Для получения кода отправьте SMS
        <b>".SMS_MESSAGE1."</b> на короткий номер
        <b>".SMS_NUMBER1."</b>
        и введите код, пришедший из ответного сообщения <br>
        <a href='javascript:void();' onclick='
            xajax_Popup_Tarif_SMS1();'>Стоимость SMS</a>
    </td>
    <td width=30% valign='top'><input type='input' name='kod_reg'
id='kod_reg' size='20' maxlength='10' value='' onchange='
xajax_Control_Reg_User(xajax.getFormValues(
\"FormRegUser\"));'></td>
    <td width=20% valign='top'>
        <div id='reg_user_kod'><font color='red'>no</font></div>
    </td></tr>";
$text1.=<tr><td></td>
    <td>
        <input type='submit' id='ButtonFormRegUser'
value='Зарегистрироваться ->' disabled=true>
    </td>
    <td></td></tr>";
$text1.=</table>";
$text1.=</form>";
```

```
return $text1;
}

////////// Всплывающее окно - тарифы ///////////
function Popup_Tarif_SMS1()
{
    $objResponse = new xajaxResponse();
    require_once("mybaza.php");
    $objResponse->assign("windowdop","style.display","none");

    $text1.=<a href='javascript:void(null);' onclick='
        document.getElementById(\"windowdop\").style.display=
        \"none\";return false;'>
        <img src='img/delete.png' align=right></a>";
    $text1.=<br><br><center>Стоимость sms 1161 для разных операторов
        </center><table>;
    // чтение из файла
    $ff1=fopen("tarif_sms_1161.csv","r");
    $text1.=<tr><td>Оператор</td><td>Цена SMS без НДС, руб.</td></tr>";
    while(!feof($ff1))
    {
        $line1=fgets($ff1,1024);
        $arrline1=explode(";", $line1);
        $text1.=<tr><td>".$arrline1[2]."</td><td>".$arrline1[5]."</td></tr>";
    }
    $text1.=</table>;

    $objResponse->script("document.getElementById(
        'windowdop').style.display='block'");
    $objResponse->assign("windowdop","innerHTML",$text1);
    $objResponse->script("document.getElementById(
        'windowdop').scrollIntoView();");

    return $objResponse;
}
```

Проверка полей формы происходит без перезагрузки страницы. Изначально кнопка **Зарегистрироваться** неактивна. При изменении поля происходит вызов AJAX-функции `xajax_Control_Reg_User` (файл `prgreg/control_reg_user.php`) с передачей параметров формы `FormRegUser`:

```
onchange='xajax_Control_Reg_User(xajax.getFormValues("FormRegUser"))'
```

E-mail проверяется на уникальность в базе. При отправке SMS через сервис `alaggregator` в базу данных (в таблицу `oplata_reg`) записывается код активации и номер телефона, с которого отправлено SMS-сообщение. Чтобы избежать повторного указания кода, при завершении регистрации в таблице `oplata_reg` изменяется

признак использования кода `active=yes` и ID нового пользователя. При попытке повторного ввода кода активации будем выводить предупреждение о том, что этот код уже использован (рис. 3.3). Отправку SMS на короткий номер через сервис a1aggregator мы рассмотрим в разд. 3.4. Функция `Control_Reg_User` находится в файле `prgreg/control_reg_user.php` (листинг 3.25).

The screenshot shows a web page with a header containing 'Логин' (Login), 'Пароль' (Password), 'Вход ->' (Login), and 'Нет сообщений' (No messages). Below the header are navigation links: Товары (Products), Заказы (Orders), Переписка (Email), Оплата (Payment), and Регистрация (Registration). On the left sidebar, there are sections for 'Заголовок' (Title), 'АКЦИЯ !!!' (Promotion), and 'Статистика' (Statistics). The promotion section shows a deal: 'База почтовых индексов населенных пунктов России 400.00 - 95.00 руб.' and 'Единый федеральный реестр туроператоров 50.00 - 45.00 руб.'. The statistics section shows data from April 22, 2010: Товаров - 10, Пользователей - 1008, Заказов - 27, and Заказов оплаченных - 15.

The main content area is titled 'Регистрация' (Registration) with the sub-instruction: 'Зарегистрируйтесь и покупайте все товары со скидкой 30 % (регистрация платная)'. It contains a 'Форма регистрации' (Registration form) with fields for 'Логин (6-12 буквы, цифры)' (Login (6-12 letters, digits)) with value 'ivanov', 'Пароль (6-12)' (Password (6-12)) with value '*****', 'Повторите пароль' (Repeat password) with value '*****', 'Ваш e-mail' (Your email) with value 'victoruni@km.ru', and 'Код активации' (Activation code) with value '123456'. To the right of these fields, a red message states: 'Такой код уже введен' (This code has already been entered), '2009-10-14' (Date), and '10:39:18' (Time). There is also a note: 'Для получения кода отправьте SMS 1+66 на короткий номер 1161 и введите код, пришедший из ответного сообщения' (To receive a code, send an SMS to 1+66 to the short number 1161 and enter the code sent in the reply message). A 'Стоимость SMS' (SMS cost) link is present. A 'Зарегистрироваться ->' (Register) button is located below the form.

To the right of the registration form is a 'Корзина' (Cart) section showing a shopping cart icon, 'Товаров - 3' (Items - 3), 'Кол-во - 5' (Quantity - 5), '50 руб' (50 rub), '1.73 usd' (1.73 usd), and '1.16 eur' (1.16 eur). A 'Подробно' (Details) link is provided. Further down on the right are sections for 'У партнеров' (With partners) and 'Контакты' (Contacts), each with their respective icons and text.

Рис. 3.3. Ошибка при повторном введении пароля при оплате через SMS-сервис

Листинг 3.25

```
// Проверка правильности заполнения полей
// при регистрации пользователя
function Control_Reg_User($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к БД
    require_once("mybaza.php");
    $query1="SELECT id FROM users WHERE login='".$utf2win($Id[login])."' ";
    $rez1=mysql_query($query1);
    $count=0;
    // проверка login - незанятость
    // цифры, буквы 6-12 символов
    if(mysql_num_rows($rez1)>0)
        $objResponse->assign("reg_user_login","innerHTML",<font
```

```
color='red'>Логин занят</font>");
elseif(!ereg("^[a-z,A-Z,a-я,A-Я,0-9]{6,12}$",$Id[login]))
    $objResponse->assign("reg_user_login","innerHTML",<font
        color='red'>ERR</font>);
else
{
    $objResponse->assign("reg_user_login","innerHTML",<font
        color='blue'>OK</font>);
    $count++;
}
// password - цифры, буквы 6-12 символов
if(!ereg("^[a-z,A-Z,a-я,A-Я,0-9]{6,12}$",$Id[password]))
    $objResponse->assign("reg_user_password","innerHTML",<font
        color='red'>ERR</font>);
else
{
    $objResponse->assign("reg_user_password","innerHTML",
        "<font color='blue'>OK</font>");
    $count++;
}
// password1 - сравнение с password
if(utf8towin($Id[password1])==utf8towin($Id[password]) &&
    strlen($Id[password1])>0)
{
    $objResponse->assign("reg_user_password1","innerHTML",<font
        color='blue'>OK</font>);
    $count++;
}
elseif(strlen(trim($Id[password1]))==0)
{
    $objResponse->assign("reg_user_password1","innerHTML",<font
        color='red'>no</font>);
    $count++;
}
else
{
    $objResponse->assign("reg_user_password1","innerHTML",<font
        color='red'><>password!</font>);
}
// email
if(ereg("^[a-z,0-9,-_\.]{2,20})([\@]{1})([a-z,0-9,-_\.]{2,20})([\.\.]
{1})([a-z,]{1,3})$",$Id[email]))
{
    $objResponse->assign("reg_user_email","innerHTML",<font
        color='blue'>OK</font>);
```

```

$count++;
}
else
$objResponse->assign("reg_user_email","innerHTML","<font
color='red'>ERR</font>");
// kod_reg
$query2="SELECT * FROM oplata_reg WHERE kod_reg='".$Id[kod_reg]."' ";
$rez2=mysql_query($query2);
$row2=mysql_fetch_assoc($rez2);
if(mysql_num_rows($rez2)==0)
$objResponse->assign("reg_user_kod","innerHTML","<font
color='red'>Неверный код</font>");
elseif($row2[activ]=='yes')
{
$message='Такой код уже введен '.$row2[data].';
$objResponse->assign("reg_user_kod","innerHTML","<font
color='red'>'.$message.'</font>");
}
else
{
$objResponse->assign("reg_user_kod","innerHTML","<font
color='blue'>OK</font>");
$count++;
}

// все поля правильно заполнены ?
if($count==5)
$objResponse->assign("ButtonFormRegUser","disabled",false);
else
$objResponse->assign("ButtonFormRegUser","disabled",true);
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}

```

После правильного заполнения всех полей и ввода кода становится активной кнопка **Зарегистрироваться**. Данные формы отправляются скрипту `ajax_Reg_User` (файл `prgreg/reg_user.php`), происходит запись данных в базу данных (таблица `users`) и отправка письма о регистрации на e-mail, указанный в форме. Вид страницы при успешной регистрации представлен на рис. 3.4. Функция `Reg_User` находится в файле `prgreg/reg_user.php` (листинг 3.26).

Листинг 3.26

```

// Регистрация пользователя
function Reg_User($Id)

```

```
{  
$objResponse = new xajaxResponse();  
$objResponse->assign("flag_ajax","value",'yes');  
// подключение к БД  
require_once("mybaza.php");  
require_once("my.php");  
$login=utftowin($Id[login]);  
$password=utftowin($Id[password]);  
$email=$Id[email];  
$data=date('Y-m-d H:i:s');  
$query1="INSERT INTO users SET data='".$data."',  
    login='".$login."',password='".$password."',  
    type='2',visible='yes',discount='".$DISCOUNT."',  
    email='".$email."',ip='".$f_get_ip()."';' "  
// Запись в базу данных  
$rez1=mysql_query($query1);  
if($rez1)  
{  
    $id=mysql_insert_id();  
    // Формирование контента для вывода  
    $text1=<center><br><br>Вы зарегистрированы!!!<br><br>  
        Информация направлена на Ваш email ".$email."<br><br>  
        <a href='javascript:void()' onclick='  
            document.getElementById(\"login\").value=\"".$login."\";  
            document.getElementById(\"password\")  
            .value=\"".$password."\";  
            document.getElementById(\"ButtonFormVhod\").disabled=true;  
            document.getElementById(\"ButtonFormVhod\").  
            value=\"Подождите...\";  
            xajax_Vhod(xajax.getFormValues(\"FormVhod\"));  
        '>Войти</a>  
    </center>;  
    // для sms кода установить признак использования  
    $query2="UPDATE oplata_reg SET id_user='".$id."',  
        data='".date('Y-m-d H:i:s')."',activ='yes'  
        WHERE kod_reg='".$Id[kod_reg]."'";  
    mysql_query($query2);  
    // отправка на e-mail  
    $to=$email;  
    $subject='Регистрация на '.SITE;  
    $body='<b>Регистрация на '.SITE.'<b><br>' ;  
    $body.='<b>Вы зарегистрированы на сайте <b>'.SITE.'<br>';  
    $body.='Данные для входа:<br>';  
    $body.='<b>Логин - <b>'.$login.'<br>';  
    $body.='<b>Пароль - <b>'.$password.'<br>';
```

```

$body.='<br>';
$body.='Администрация сайта '.SITE.'<br>';
$body.='email - <a href=mailto:'.$EMAILADMIN.''.
    '>'.$EMAILADMIN.'</a><br>';
$headers="Content-type: text/html; charset=windows-1251;";
mail($to,$subject,$body,$headers);
}
else
{
$text1=<center><br><br>Ошибка регистрации!!!<br>
<br>Обратитесь к администратору</center>";
}
$objResponse->assign("center2","innerHTML",$text1);
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}

```

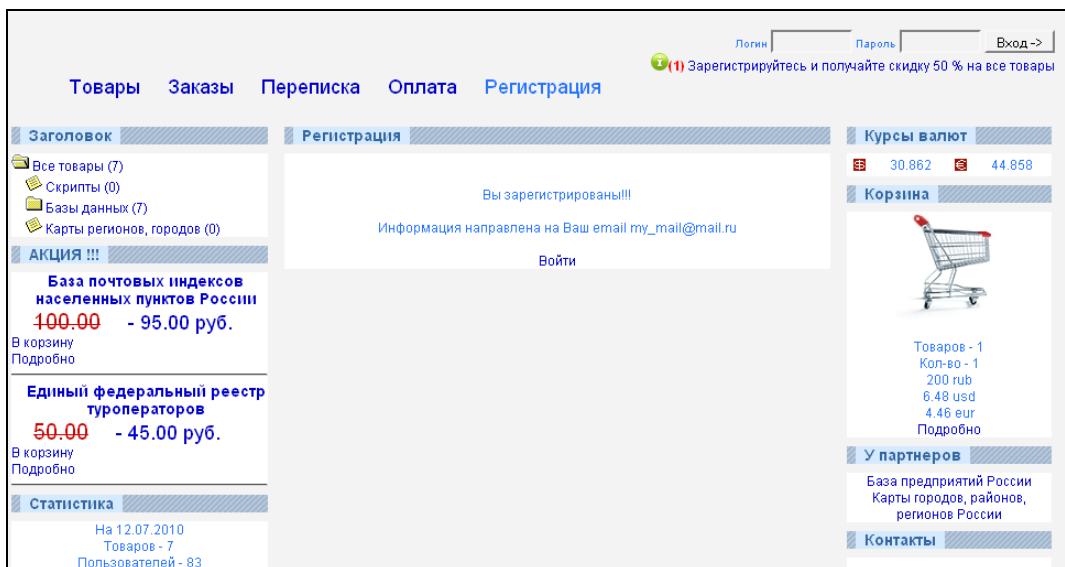


Рис. 3.4. Пользователь зарегистрирован

При нажатии на кнопку **Войти** пользователь попадает на сайт под своим профилем.

3.4. Оплата SMS через сервис a1aggregator

Для приема SMS-платежей зарегистрируемся в сервисе a1aggregator.ru. Зайдите по адресу <http://www.a1aggregator.ru> и далее по ссылке **Регистрация**. Зарегистрироваться и стать партнером вы можете, заполнив форму, приведенную на рис. 3.5.

Регистрация

Логин:

Пароль:

Подтвердите пароль:

E-mail:

Номер телефона:

Откуда вы о нас узнали:

Зарегистрироваться или [Отменить](#)

Рис. 3.5. Форма регистрации в сервисе a1aggregator

Инструменты

- ▶ [A1Lite](#)
- ▼ [SMS API](#)
 - [A1 API](#)
 - [A1 EASY](#)
 - [A1 Script](#)
 - [A1 Sharp](#)
 - [Статистика](#)
 - [Тарифы](#)
 - [Библиотека](#)
 - [Справочник](#)
 - [SOAP](#)
- ▶ [WebMoney API](#)

A1 API

С помощью A1 API вы работаете напрямую с платформой. Платеж контролируется с помощью скрипта-обработчика на вашей стороне. Для начала работы добавьте новый сервис и настройте его параметры. Дополнительную информацию по работе с A1 API вы можете [узнать в нашей базе знаний \(wiki\)](#)

[+ Добавить сервис](#)

Название	Тарифный план	Статус	Действия
Оплата SMS	Стандартный	Новый	Удалить
Online-spravka	Стандартный	Новый	Удалить

Отчеты

- [A1Lite](#)
- [SMS](#)
- [WebMoney](#)
- [Уведомления](#)

Финансы

Рис. 3.6. Создание нового платежного сервиса

При регистрации указывайте существующий номер телефона, т. к. при выводе платежей на этот номер приходит код подтверждения. После регистрации вам будет присвоен анонимный аттестат и вы сможете протестировать работу сервисов. Для снятия всех ограничений анонимного аттестата и запуска сервисов в "боевом" режиме вам необходимо будет пройти процедуру получения формального аттестата.

В дальнейшем вы также можете получить персональный аттестат и VIP-аттестат. После заполнения регистрационных данных и нажатия кнопки **Сохранить** вы автоматически попадаете в панель управления. Теперь вы сможете создать свой платежный сервис (вкладка **A1 API**). Откройте вкладку **A1 API** и нажмите ссылку **Добавить сервис** (рис. 3.6).

Укажите название нового сервиса, а также адрес скрипта-обработчика на вашем сайте (рис. 3.7).

A1 Sharp

Статистика

Тарифы

Библиотека

Справочник

SOAP

▶ **WebMoney API**

Отчеты

A1Lite

SMS

WebMoney

Уведомления

Финансы

Счета

Заявки

Помощь

Поддержка

WIKI

Форум

● Стань агентом **a1a**

Параметры

Название: Goodtovars

URL обработчика: http://goodtovars.ru/prgreq/sms.php

Дополнительный URL:

Вы можете указать адрес дополнительного скрипта-обработчика: он будет вызываться, если основной по какой-либо причине не ответил.

Адрес службы поддержки: http://goodtovars.ru/

Укажите адрес службы поддержки - корневой домен второго или третьего уровня (без учета "http://" и "www" и без дополнительных страниц "page.php" и директорий "dir"). Например: a1help.ru

Сохранить

© 2010 "A1-Aгрегатор"

Рис. 3.7. Заполнение реквизитов нового платежного сервиса

Вы можете указать адрес дополнительного скрипта-обработчика: он будет вызываться, если основной по какой-либо причине не ответил. Нажмите кнопку **Сохранить**. Ваш первый платежный сервис готов. После создания платежного сервиса вы автоматически попадаете на страницу редактирования его параметров (рис. 3.8). Еще раз проверьте, верно ли вы указали данные. Теперь вам необходимо выбрать один или несколько префиксов для вашего нового сервиса. Префикс — это текст, который передается в SMS-сообщении: по нему определяется, кому именно принадлежит платежный сервис. Вы выбираете префикс первого уровня, имеющийся в системе, а затем сами прописываете к нему префикс второго уровня (не менее трех символов). В сумме префиксы первого и второго уровня составляют префикс, который отправляет абонент. Помимо префикса вы для повышения безопасности можете указать секретный ключ (делать это необязательно). Секретный ключ — это последовательность символов, которая кодируется по алгоритму MD5

и передается обработчику при помощи GET-запроса. Вновь попасть на страницу редактирования сервиса вы сможете, нажав на название нужного сервиса во вкладке **A1 API**. После указания всех необходимых параметров вы можете протестировать созданный сервис. В разделе тестирования сервиса укажите номер мобильного телефона абонента, оператора мобильной связи, короткий номер и текст сообщения. Система проведет имитацию отправки сообщения абонентом и передаст все данные вашему скрипту-обработчику, после чего в специальном окне вы сможете просмотреть ответ вашего сервиса.

Email	Тарифный план	Стандартный изменить
SMS	Схема выплат	95.00%
WebMoney	Ресурс сервиса	Goodtovars
Уведомления	URL обработчика	http://goodtovars.ru/prgreg/sms.php
Финансы	Дополнительный URL	 Вы можете указать адрес дополнительного скрипта-обработчика: он будет вызываться, если основной по какой-либо причине не ответил.
Счета	Адрес службы поддержки	index.php
Заявки	Укажите адрес службы поддержки - корневой домен второго или третьего уровня (без учета "http://" и "www" и без дополнительных страниц "page.php" и директорий "dir"). Например: a1help.ru	
Помощь	Секретный ключ	 Секретный ключ – это последовательность символов, которая кодируется по алгоритму MD5, и передается обработчику при помощи GET-запроса. Применяется в целях дополнительной безопасности. Указывать секретный ключ не обязательно.
Поддержка	Префиксы	<input checked="" type="checkbox"/> 1+regis <input checked="" type="checkbox"/> 1+regs
WIKI	Добавить префикс	# <input type="button" value="▼"/> <input type="text"/>
Форум		

Рис. 3.8. Заполнение реквизитов нового платежного сервиса

После отправки пользователем SMS-сообщения на короткий номер, сервис a1aggregator отправляет запрос на адрес, указанный в URL обработчика (в данном случае на адрес <http://goodtovars.ru/prgreg/sms.php>). Содержимое файла prgreg/sms.php приведено в листинге 3.27.

Примерный вид запроса:

```
http://goodtovars.ru/prgreg/sms.php?user_id=79283456789&num=1161&cost=100&cost_rur=100&msg=1%2Bregs&skey=d41d8cd98f00b204e9800998ecf8427e&operator_id=112&date=2010-04-22+12%3A10%3A05&smsid=1282022226&msg_trans=1%2Bregs&operator=operator&test=1&ran=5&try=1&country_id=4846&sign=74fb9b1b1535ae4ee02a606b541f1812
```

Здесь:

- user_id — номер телефона отправителя SMS;
- num — короткий номер (1161);
- msg — сообщение (1+reg);

- cost — сумма платежа;
- smsid — ID платежа в системе a1agregator (1282022226).

После получения запроса URL-обработчик должен дать один из возможных ответов:
 smsid:1282022226\nstatus:ignore\n";

или

smsid1282022226\nstatus:reply\nkod:Пароль\nСообщение\n

Здесь:

- Пароль — код для входа;
- Сообщение — сообщение, которое будет отправлено на телефон отправителя, естественно, содержащее код для входа.

Листинг 3.27

```
<?
// подключить файл настроек
require_once("../my.php");
// подключение к бд
require_once("../mybaza.php");
// проверить короткий номер и текст сообщения
if( $_GET['num'] !=SMS_NUMBER1 || $_GET['msg_trans'] !=SMS_MESSAGE1 )
{
    // неверно
    $smsid = $_GET['smsid'];
    // ответ отрицательный
    echo "smsid:$smsid\n";
    echo "status:ignore\n";
}
// ответ положительный
// все нормально - записать номер телефона и код в базу
else
{
    $password= $_GET['smsid'];
    $smsid = $_GET['smsid'];
    echo "smsid:$smsid\n";
    echo "status:reply\n";
    echo "kod:$password\n";
    echo "\n";
    echo "Usluga oplachena.Kod=". $password ."\n";
    $query1="INSERT INTO oplata_reg SET data='".date("Y-m-d H:i:s")."',kod_"
    reg='".$_. $_GET['smsid']."' ,
        phone='".$_. $_GET['user_id']."' ,activ='no' ";
    $rez1=mysql_query($query1);
}
?>
```

3.5. Блок "Товары"

Блок "Товары" должен содержать:

- список категорий товаров неограниченной вложенности;
- форму поиска товара;
- список товаров категории или результатов поиска постранично;
- подробный показ товара;
- корзину неограниченной вложенности.

Положить товар в корзину можно нажатием на ссылку либо перетаскиванием картинки товара. Корзина показывает общее количество товаров, наименования всех товаров в корзине и их стоимость в трех валютах. Корзину можно просмотреть и изменить подробно (по ссылке **Подробно**). Из формы **Корзина подробно** можно оформить заказ.

3.5.1. Список категорий товаров неограниченной вложенности

Наша задача — создать список категорий товаров неограниченной вложенности. В БД за формирование списка отвечает таблица `kategory`. Напомним ее структуру:

- `id` — первичный ключ;
- `id_parent` — ID родительской категории (0 — верхняя);
- `name` — название категории;
- `nn` — количество товаров во всех дочерних категориях;
- `visible` — статус (yes — показывать; no — не показывать).

Вид дерева категорий иллюстрирует рис. 3.9.

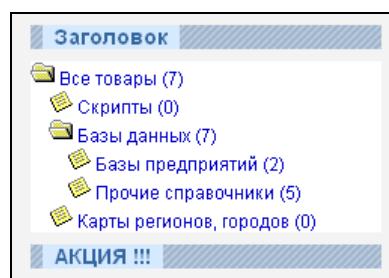


Рис. 3.9. Вид дерева категорий

При нажатии на ссылку (иконку или название) происходит выполнение соответствующей подпрограммы. Для раскрытой категории — свертывание (`ajax_Close_Kategory`); для закрытой — раскрытие (`ajax_Open_Kategory`); поиск вложенных категорий и формирование ссылок; для нижних — вывод списка товаров данной категории (`ajax_View_Tovars_Kategory`); т. к. вывод товаров происходит

дит постранично, в передаваемых параметрах указываем первую страницу. Для каждой категории контент выводится в блок `id=kategoryN` (`id` — ID данной категории в таблице базы данных). В итоге блоки с контентом располагаются так, как показано на рис. 3.10.

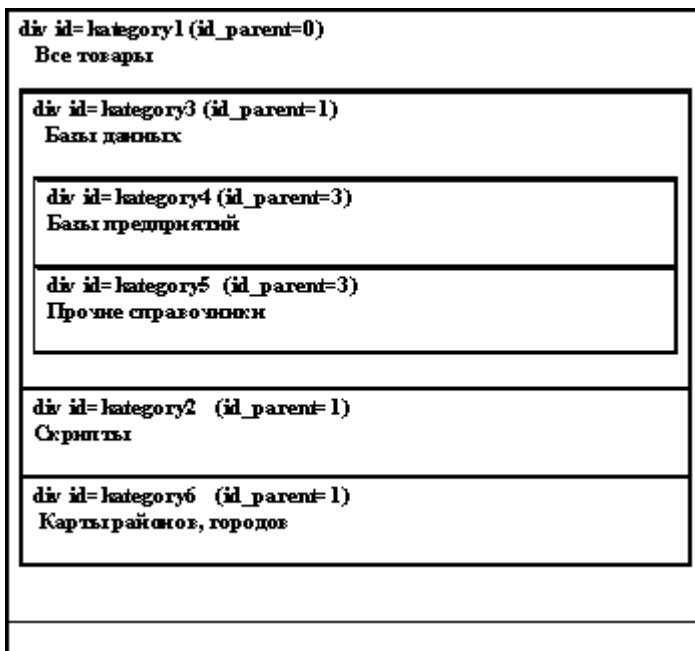


Рис. 3.10. Расположение блоков с контентом в дереве категорий

Вид (и перенаправление при нажатии) иконки для категории товаров зависит от наличия вложений. Поэтому при раскрытии каждой вложенной категории необходим поиск вложенных. Функции, отвечающие за развертывание (свертывание), находятся в файле `prgtovars/open_close_kategory.php` (листинг 3.28).

Листинг 3.28

```

// Разворачивание категории
function Open_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // получение вложенных категорий
    $content=f_open_kategory($Id);
    // вывести в блок
    $objResponse->assign("category".$Id,"innerHTML",$content);
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}

```

```
    }
    // свертывание категории
    function Close_Kategory($Id)
    {
        $objResponse = new xajaxResponse();
        $objResponse->assign("flag_ajax","value",'yes');
        // убрать вложенные категории
        $content=f_close_kategory($Id);
        // вывести в блок
        $objResponse->assign("kategory".$Id,"innerHTML",$content);
        $objResponse->assign("flag_ajax","value",'no');
        return $objResponse;
    }
```

Функции, формирующие контент, находятся в файле prgtovars/function_open_close_category.php (листинг 3.29).

Листинг 3.29

```
/// Выдача дерева категорий
// показать вложенные подкаталоги
function f_open_category($Id)
{
    require_once("mybaza.php");
    $text1="";
    // получение списка вложенных категорий
    $query1="SELECT name,id,nn FROM kategory WHERE id='".$Id."'"
           && visible='yes' ";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    $text1.="
```

```

{
    $text1.=<div class='menu' id='kategory".$row2[id]."'>
        <span><a href='javascript:void();' onclick=
            xajax_Open_Kategory(".$row2[id].")'>
            <img src='img/open_dir.ico'></a></span>
        <span><a href='javascript:void();' onclick=
            xajax_Open_Kategory(".$row2[id].")'>
            ".$row2[name]." (".$row2[nn].")</a><span></div>";
    }
}

// категории конечные
else
{
    $text1.=<div class='menu' id='kategory".$row2[id]."'>
        <a href='javascript:void();' onclick=
            var x=new Array();x[0]='".$row2[id].",x[1]=1;
            xajax_View_Tovars_Kategory(x)'>
            <img src='img/last_dir.ico'></a>
        <a href='javascript:void();' onclick=
            var x=new Array();x[0]='".$row2[id].",x[1]=1;
            xajax_View_Tovars_Kategory(x)'>
            ".$row2[name]." (".$row2[nn].")</a></div>";
    }

}

return $text1;
}

// свертывание категории
function f_close_kategory($Id)
{
    require_once("mybaza.php");
    $text1="";
    // получение списка вложенных категорий
    $query1="SELECT name,id,nn FROM kategory WHERE id='".$$.Id."'"
        && visible='yes' ";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    $text1.=<a href='javascript:void();' onclick=
        xajax_Open_Kategory(".$Id.")'>
        <img src='img/open_dir.ico'></a>
        <a href='javascript:void();' onclick=
            xajax_Open_Kategory(".$Id.")'>
            ".$row1[name]." (".$row1[nn].")</a>";

    return $text1;
}

```

3.5.2. Вывод списка товаров постранично

При выборе нижней категории в списке категорий в блок center2 подгружается список товаров этой категории постранично (рис. 3.11). О каждом товаре будем выводить следующую информацию:

- название;
- картинку;
- краткое описание;
- обычную цену;
- специальную цену (если есть); при этом обычная цена выводится зачеркнутой;
- персональную скидку (если есть);
- ссылку **Подробно** (посмотреть товар подробно);
- ссылку **В корзину**;
- ссылку **Редактировать** (для администратора).

Функция `View_Tovars_Kategory`, отвечающая за вывод товаров категории, находится в файле `prgtovars/view_tovars_kategory.php` (листинг 3.30). Входные параметры — `id` выбранной категории и номер страницы вывода.

Листинг 3.30

```
<?php
function View_Tovars_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // заголовок - путь к текущей категории
    $zag=f_zag1(f_string_kategory($Id[0]));
    $objResponse->assign("centercaption3","innerHTML",$zag);
    // формирование контента товаров текущей страницы
    // и ссылок для переходов по страницам
    $content=f_view_tovars_kategory($Id);
    $objResponse->assign("center3","innerHTML",$content[0]);
    $objResponse->assign("center4","innerHTML",$content[1]);
    // перенести div center3 в зону видимости
    $objResponse->script("document.getElementById
        ('center3').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Функция `view_Tovars_Kategory` (из файла `prgtovars/view_tovars_category.php`) записывает результаты выполнения в соответствующие блоки:

- center2 — заголовок (путь до категории);
- center3 — текущую страницу с товарами (краткое описание);
- center4 — ссылки на другие страницы.

Вывод списка товаров категории иллюстрирует рис. 3.11. Функции, формирующие контент, находятся в файле `prgtovars/function_view_tovars_category.php` (листинг 3.31).

Рис. 3.11. Вывод списка товаров категории

Листинг 3.31

```
<?php
// Просмотр товаров категории постранично
// Входящие данные
// $Id[0] - id категории
// $Id[1] - номер страницы для показа
function f_view_tovars_category($Id)
{
    // файл настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
```

```
// подключение файла с подпрограммой формирования пути к текущей
// категории
require_once("prgtovars/function_string_kategory.php");
// сначала подсчитаем общее количество товаров категории
$query0="SELECT COUNT(id) FROM tovars WHERE
        id_kategory='".$Id[0]."' && visible='yes' ";
$rez0=mysql_query($query0);
$count=mysql_result($rez0,0);
// потом номер страницы вывода и номер первого элемента вывода из БД
// количество товаров на страницу - константа NN1 (в файле my.php)
$pages=ceil($count/NN1);
$page=min($Id[1],$pages);
$poz=($page-1)*NN1;
// если общее количество товаров ($count>0), продолжим.
// в противном случае $text=' По данному запросу поиска ничего не обнаружено ';
if($count>0)
{
    // запрос товаров для нужной страницы
    // из базы данных числом NN1, начиная с poz
    $query1="SELECT * FROM tovars WHERE id_kategory='".$Id[0]."'
        && visible='yes' LIMIT ".$poz.", ".NN1."'";
    $rez1=mysql_query($query1);
    while($row1=mysql_fetch_assoc($rez1))
    {
        // наименование товара
        $text1.= "<div class='zag_tovar'><span
            onmousedown='var x=document.getElementById(\"windowdrag\");
            x.style.visibility=\"visible\";
            var y=\"".$row1[name]."\";x.innerHTML=y;
            flag1.status=1;flag1.id=\".$row1[id].\";return false;'>
            ".$row1[name]."</span></div>";
        $text1.= "<div class='kategory_tovar'>".f_string_kategory
            ($row1[id_kategory])."</div>";
        $text1.= "<div class='info_tovar'>";
        // картинка уменьшенная
        $text1.= "<table><tr><td width=20%><img src='resize_100.php?
            pic=".$row1[img]."' onmousedown='var x=document.getElementById
            (\\"windowdrag\\");x.style.visibility=\"visible\";
            var y=\"<img src=resize_100.php?pic=".$row1[img].">
            \";x.innerHTML=y;
            flag1.status=1;flag1.id=\".$row1[id].\";return false;'></td>";
        // инфо о товаре
        $text1.= "<td width=80%>".$row1[info]."</td></tr></table>";
        $text1.= "</div>";
        // есть цена по акции - выводим основную зачеркнутой
```

```
// и цену по акции
if($row1[new_pay_rub]>0)
{
    $text1.= "<div class='no_pay_tovar'>Цена - ".$row1[pay_rub]."
руб.</div>";
    $text1.= "<div class='new_pay_tovar'>Спец. цена - "
    ".$row1[new_pay_rub]." руб.</div>";
}
// нет цены по акции - выводим только основную
else
    $text1.= "<div class='pay_tovar'>Цена - ".$row1[pay_rub]."
руб.</div>";
// скидка клиента
$query2="SELECT discount FROM users WHERE
id='".$SESSION[user].'";
$text1.= "<div class='discount_tovar'>Скидка - ".
mysql_result(mysql_query($query2),0)." %</div>";
// ссылки в Корзину и Товар Подробно
$text1.= "<div>
    <div class='function_tovar'>
        <a href='javascript:void()' onclick='
            xajax_Add_To_Korzina(\".$row1[id].\");
        '>
            В корзину</a>
            (или drag&drop img или названия)</div>
        <div class='function_tovar'>
            <a href='javascript:void()' onclick='
                xajax_View_Tovar(\".$row1[id].\");
            '>
                Подробно</a>
        </div>";
// для админа - Редактировать товар
if($_SESSION[type]>7)
    $text1.= "<div class='function_tovar'>
        <a href='javascript:void()' onclick='
            xajax_Edit_Tovar(\".$row1[id].\");
        '>
            Редактировать</a>
    </div>";
    $text1.= "<div class='linedown_tovar'><hr></hr></div>";
    $text1.= "</div>";
}
// ссылка для админа на добавление нового товара
if($_SESSION[type]>7)
    $text1.= "<div class='function_tovar'>
        <a href='javascript:void()' onclick='
            xajax_Add_New_Tovar();
        '>Добавить новый товар</a>
    </div>";
```

```
// список ссылок перехода по страницам
$text2="";
if($pages>1)
{
if($page != 1)
{$i=$page-1;
$text2.="
```

```
// возврат контента - страницы
$text[1] = $text2;
return $text;
}
?>
```

Число товаров на странице (константа `NN1`) находится в файле `my.php` (`define (NN1, 5);`).

ЗАМЕЧАНИЕ

Здесь следует обратить внимание на вывод картинки. Для сокращения трафика и ускорения вывода на сервере формируется и передается пользователю уменьшенная картинка ``. Файл формируется с помощью стандартной библиотеки PHP — GD2.

В зависимости от типа пользователя, для каждого товара есть ссылки на подробный просмотр, редактирование и отправку товара в корзину. Поместить товар в корзину можно также перетаскиванием картинки (или наименования) товара на корзину.

3.5.3. Динамический "ресайзер" картинок

Динамический "ресайзер" — это программа вывода картинок с сохранением пропорций и максимальным размером 100 px по горизонтали или вертикали. Для ее работы необходимо наличие стандартной PHP-библиотеки GD2. Для экономии трафика "ресайз" осуществляется на стороне сервера. Программа находится в файле `resize_100.php` (листинг 3.32).

Листинг 3.32

```
<?php
if($_GET['pic'])
{
$img = new img($_GET['pic']);
$img->resize();
$img->show();
}

class img {
var $image = '';
var $temp = '';
function img($sourceFile)
{
if(file_exists($sourceFile))
{
$pictype= strrchr($sourceFile,".");
if ($pictype==".png")
$this->image = ImageCreateFromPNG($sourceFile);
```

```
if ($pictype=="gif")
    $this->image = ImageCreateFromGIF($sourceFile);
if ($pictype=="jpg")
    $this->image = ImageCreateFromJPEG($sourceFile);
}
else
{
    $this->errorHandler();
}
return;
}
// resize
function resize($width = 100, $height = 100, $aspectradio = true)
{
    $o_wd = imagesx($this->image);
    $o_ht = imagesy($this->image);
    if(isset($aspectradio) &&$aspectradio)
    {
        $w = round($o_wd * $height / $o_ht);
        $h = round($o_ht * $width / $o_wd);
        if(($height-$h)<($width-$w))
        {
            $width =& $w;
        }
    else
        {
            $height =& $h;
        }
    }
    $this->temp = imageCreateTrueColor($width,$height);
    imageCopyResampled($this->temp, $this->image,
    0, 0, 0, 0, $width, $height, $o_wd, $o_ht);
    $this->sync();
    return;
}
function sync()
{
    $this->image =& $this->temp;
    unset($this->temp);
    $this->temp = '';
    return;
}
function show()
{
    $this->_sendHeader();
```

```

ImageJPEG($this->image);
return;
}

function _sendHeader()
{
header('Content-Type: image/jpeg');
}

function errorHandler()
{
echo "error";
exit();
}

function store($file)
{
ImageJPEG($this->image,$file);
return;
}

function watermark($pngImage, $left = 0, $top = 0)
{
ImageAlphaBlending($this->image, true);
$layer = ImageCreateFromPNG($pngImage);
$logoW = ImageSX($layer);
$logoH = ImageSY($layer);
ImageCopy($this->image, $layer, $left, $top, 0, 0, $logoW, $logoH);
}
}

?>
```

3.5.4. Программирование навигатора страниц

Количество товаров или заказов в магазине может быть очень большим. При выводе товаров (или заказов) постранично переход на нужные страницы осуществляется по ссылкам, расположенным в блоке ниже списка товаров (заказов) текущей страницы. Вид навигатора иллюстрируют рис. 3.12 и 3.13. Если страниц больше 20, список не будет помещаться в одну строку, что очень неудобно. Для удобства навигации по страницам создадим навигатор страниц вида

```
< 1 2 3 .... n-2 n-1 n n+1 n+2 .... last-2 last-1 last >
```

т. е. предыдущая страница, n первых, n средних, n последних и последующая страница (n — настраиваемый параметр). Для реализации навигатора напишем функцию `dopfun`.

Для вывода блока ссылок для перехода по страницам напишем функцию `doarray1` (файл `dopfun.php`). Задача процедуры — выдать массив вида

```
[1,2,3,...,7,8,9,...,13,14,15] .
```

```
|__k_|
```

Номер	Логин	Дата	Сумма руб	Оплата руб	Статус			
29	11111	2010-07-18 13:07:22	117.50	0.00	нет			
28	280a865fce5c...	2009-11-16 12:59:57	210.00	0.00	нет			
27	280a865fce5c...	2009-11-05 11:01:22	10.00	0.00	нет			
26	280a865fce5c...	2009-11-02 17:45:18	210.00	0.00	нет			
25	280a865fce5c...	2009-11-02 17:41:10	10.00	0.00	нет			
24	280a865fce5c...	2009-11-02 17:39:25	10.00	0.00	нет			
23	280a865fce5c...	2009-11-02 17:38:10	10.00	0.00	нет			
22	280a865fce5c...	2009-11-02 17:36:47	45.00	0.00	нет			
21	280a865fce5c...	2009-11-02 17:23:38	10.00	0.00	нет			
20	280a865fce5c...	2009-11-02 17:22:41	45.00	0.00	нет			

1 2 3 >>

Всего - 29 Страниц - 3

Рис. 3.12. Вид навигатора страниц

Id	Дата	Логин	руб	Тип	Статус	ip			
153	2009-11-03 06:41:38	dc8977e52055...		Гость	yes	75.101.140.241			
154	2009-11-03 16:40:22	f932be9ba95...		Гость	yes	78.107.69.114			
155	2009-11-03 17:05:10	f8ca10ea8ab39...		Гость	yes	109.188.24.178			
156	2009-11-03 18:13:08	5f3a36ab2e3a...		Гость	yes	81.25.58.23			
157	2009-11-03 18:13:13	2259b01efa0b...		Гость	yes	81.25.58.23			
158	2009-11-03 22:32:37	0a7eccf429eb...		Гость	yes	77.248.237.185			
159	2009-11-03 23:27:36	72c1d55070b8...		Гость	yes	81.90.9.71			
160	2009-11-04 09:12:53	e00edb3c301c...		Гость	yes	89.105.136.3			
161	2009-11-04 11:31:35	190025270e2d...		Гость	yes	90.150.119.227			
162	2009-11-04 18:36:12	7d50ebdc9a4f...		Гость	yes	81.25.58.23			

<< 1 2 ... 12 13 14 15 16 17 ... 165 166 >>

Всего - 1660 Страниц - 166

Рис. 3.13. Вид навигатора при большом количестве страниц

Функции dopfun и doarray1 находятся в файле dopfun.php (листинг 3.33).

Листинг 3.33

```
// Страны страниц
//1,2..k1,k2,k3,k4..n-1,n
// arg1 - тек. страница
// arg2 - кол-во страниц
// arg3 - ширина блока k
function doarray1($arg1,$arg2,$arg3)
{
$y=array();
for($i=$arg1;$i<$arg2;$i++)
{
$y[]=$i;
}
return $y;
}
```

```

if($arg2<=($arg3+4)) // без точек
{
    for($j=0;$j<$arg2;$j++)
        $y[$j]=$j+1;
}
else
{
    if($arg1<=$arg3+2) // 11121....11111
    {
        for($j=0;$j<($arg3+2);$j++)
            $y[$j]=$j+1;
        $y[$j]="...";$j++;
        $y[$j]=$arg2-1;$j++;
        $y[$j]=$arg2;
    }
    elseif($arg1>($arg2-$arg3-2)) // 1111.....11211
    {
        $y[0]=1;
        $y[1]=2;
        $y[2]="...";
        for($j=3;$j<($arg3+5);$j++)
            $y[$j]=$arg2-$arg3+$j-4;
    }
    else // 11..112111...11
    {
        $y[0]=1;
        $y[1]=2;
        $y[2]="...";
        for($j=3;$j<=($arg3+3);$j++)
            $y[$j]=$arg1+$j-4;
        $y[$j]="...";$j++;
        $y[$j]=$arg2-1;$j++;
        $y[$j]=$arg2;
    }
}
return $y;
}

```

3.5.5. Вывод пути к категории товаров

При выводе товара отображается его месторасположение в дереве категорий — полный путь от родительской категории до текущей (рис. 3.14). Функция формирования пути `f_string_category` находится в файле `prgtovar/function_string_category.php` (листинг 3.34).

Листинг 3.34

```
<?php
// Создание строки пути категории
function f_string_kategory($Id)
{
    // файл настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    $text1="";
    $query0="SELECT * FROM kategory WHERE id='".$Id."'";
    $rez0=mysql_query($query0);
    $row0=mysql_fetch_assoc($rez0);
    $text1=$text1.$row0[name];
    $id_parent=$row0[id_parent];
    // пока есть родительская категория
    while($id_parent>0)
    {
        $query0="SELECT * FROM kategory WHERE id='".$id_parent."'";
        $rez0=mysql_query($query0);
        $row0=mysql_fetch_assoc($rez0);
        $text1=$row0[name]."->".$text1;
        $id_parent=$row0[id_parent];
    }
    return $text1;
}
?>
```



Рис. 3.14. Вывод полного пути к категории товара

Алгоритм формирования пути такой: берем входные данные ID текущей категории, находим ее наименование, записываем в текстовую переменную и движемся вверх по дереву, пока не достигнем корня, при этом название категории с разделителем → дописывается слева в текстовую переменную.

3.5.6. Поиск товаров и вывод постранично

Вид формы поиска товаров независимо от нахождения в категории формируется в файле function_form_search_tovars.php. Поиск ведется по соответствуанию названия или описания товара (рис. 3.15). Форме присваивается id = FormSearchTovars. В скрытом поле передается номер страницы вывода результатов поиска:

```
input type='hidden' id='pagesearch' name='pagesearch' value='1'>
```

Форма поиска полного товара (рис. 3.15) имеет следующую структуру:

- Название**: поле для ввода текста.
- Поиск в**: поле для выбора критерия поиска:
 - в названии
 - в описании
- Найти ->**: кнопка поиска.

Рис. 3.15. Форма поиска полного товара

Функция формирования формы поиска f_form_search_tovar находится в файле prgtovars/function_form_search_tovar.php (листинг 3.35).

Листинг 3.35

```
<?php
// Форма поиска товара
function f_form_search_tovars()
{
    // файл настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    // создание формы поиска
    $text1.=<form id='FormSearchTovars' action='javascript:void(null);'
        onsubmit='xajax.$(\"ButtonFormSearchTovars\").disabled=true;
        xajax.$(\"ButtonFormSearchTovars\").value=\"Подождите...\";
        xajax_View_Search_Tovars(xajax.getFormValues(
            \"FormSearchTovars\"));\'>;
    $text1.=<table width=100%>;
    $text1.=<tr><td width=50%>Название</td>;
    $text1.=<td width=50%>
        <input type='text' name='name' value='' size=20 maxlength=20>
        <input type='hidden' id='pagesearch' name='pagesearch'
            value='1'></td></tr>;
    $text1.=<tr><td width=50%>Поиск в </td>;
    $text1.=<td width=50%>
```

```

<input type='checkbox' name='inname' value='1' checked>
    в названии
<br><input type='checkbox' name='ininfo' value='1' > в описании
</td></tr></table>";
$text1.= "<center><input type='submit' id='ButtonFormSearchTovars'
    value='Найти ->'></center>";
$text1.= "</form>";
return $text1;
}
?>

```

Данные формы отправляются в функцию `xajax_View_Search_Tovars`. Функция `View_Search_Tovars` (из файла `prgtovars/view_tsearch_tovars.php`) записывает результаты выполнения в следующие блоки:

- center2 — заголовок (путь до категории);
- center3 — текущую страницу с товарами (краткое описание);
- center4 — ссылки на другие страницы.

Вывод результата поиска товаров иллюстрирует рис. 3.16.

Рис. 3.16. Вывод результатов поиска товаров

Функция `View_Search_Tovars`, отвечающая за вывод товаров категории, находится в файле `prgtovars/view_search_tovars.php` (листинг 3.36). Входные параметры — ID выбранной категории и номер страницы вывода.

Листинг 3.36

```

<?php
// Просмотр товаров по поиску
function View_Search_Tovars($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // вывести заголовок
    $zag=f_zag1("Результат поиска");
    $objResponse->assign("centercaption3","innerHTML",$zag);
    // получить результаты поиска
    $content=f_view_search_tovars($Id);
    // вывести страницу поиска
    $objResponse->assign("center3","innerHTML",$content[0]);
    // вывести навигатор страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
    // блок center3 в зону видимости
    $objResponse->script("document.getElementById
        ('center3').scrollIntoView();");
    // активировать кнопку
    $objResponse->assign("ButtonFormSearchTovars","value", "Найти ->");
    $objResponse->assign("ButtonFormSearchTovars","disabled", false);

    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>

```

Функция `f_view_search_tovars`, формирующая контент, находится в файле `prgtovars/function_view_search_tovars.php` (листинг 3.37). Алгоритм аналогичен поиску товаров категории. Сначала подсчитаем общее количество товаров. Потом номер страницы вывода и номер первого элемента вывода. Число товаров на странице — константа `NN1` в файле `my.php`. Если общее число товаров больше нуля (`$count>0`), продолжим. В противном случае выводим сообщение **По данному запросу поиска ничего не обнаружено** (рис. 3.17). Отличие в самом SQL-запросе. Для поиска товаров в SQL-запросе с помощью оператора `WHERE` вводятся условия для выбора позиций, удовлетворяющих условиям поиска.

Поля формы используются в качестве фильтра, при пустых полях выводится весь список товаров. Еще обратите внимание на оператор `LIKE %string%` в SQL-запросе.

О каждом товаре будем выводить следующую информацию:

- название;
- картинку;

- краткое описание;
 - обычную цену;
 - специальную цену (если есть); при этом обычная цена выводится зачеркнутой;
 - персональную скидку (если есть);
 - ссылку **Подробно** (посмотреть товар подробно);
 - ссылку **В корзину**;
 - ссылку **Редактировать** (для администратора).

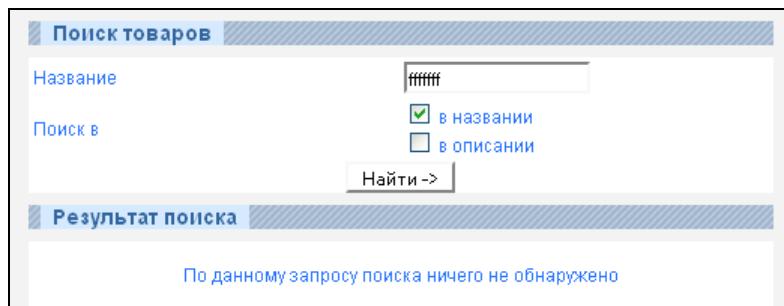


Рис. 3.17. Пустой результат поиска товаров

Листинг 3.37

```
<?php
function f_view_search_tovars($Id)
{
// файл настроек
require_once("my.php");
// подключение к базе данных
require_once("mybaza.php");
// подключение файла с подпрограммой формирования пути к текущей
//категории
require_once("prgtovars/function_string_kategory.php");
$text=array();
$text1="";
// формирование запроса
$query0="SELECT COUNT(id) FROM tovars WHERE visible='yes' ";
$name=utf8towin($Id[name]);
if(strlen(rtrim(ltrim($name)))>0)
{
if($Id[inname]==1 && $Id[ininfo]==1)
$query0.="&& (name LIKE '%".$name."%' || info LIKE '%".$name."%') ";
elseif($Id[inname]==1)
$query0.="&& name LIKE '%".$name."%' ";
```

```
elseif($Id[ininfo]==1)
$query0.=="&& info LIKE '%".$name."%' ";
else
;
}
$rez0=mysql_query($query0);
// получить количество записей в результате запроса
$count=mysql_result($rez0,0);
// получить количество страниц
$pages=ceil($count/NN1);
// корректировка количества страниц
$page=min($Id[pagesearch],$pages);$poz=($page-1)*NN1;
$text1=<div class='zag_view_tovars'>";
// результат запроса непустой
if($count>0)
{
$query0.=" LIMIT ".$poz.", ".NN1."'";
$query1=str_replace("COUNT(id)","*", $query0);
$rez1=mysql_query($query1);
while($row1=mysql_fetch_assoc($rez1))
{
// наименование товара
$text1.= "<div class='zag_tovar'><span onmousedown='
var x=document.getElementById(\"windowdrag\");
x.style.visibility=\"visible\";
var y=\"$".$row1[name]."\";
x.innerHTML=y;
flag1.status=1;flag1.id=\"$row1[id]\";
return false;'>".
">$".$row1[name]."</span></div>";
// путь к категории товара
$text1.= "<div class='kategory_tovar'>".f_string_kategory(
$row1[id_kategory])."</div>";
$text1.= "<div class='info_tovar'>";
// картинка товара
$text1.= "<table><tr><td width=20%><img src='resize_100.php?
pic=".$row1[img]."' onmousedown='
var x=document.getElementById(\"windowdrag\");
x.style.visibility=\"visible\";
var y=<img src=resize_100.php? pic=".$row1[img].">\";
x.innerHTML=y;flag1.status=1; flag1.id=\"$row1[id]\";
return false;'></td>";
// краткая информация о товаре
$text1.= "<td width=80%>".$row1[info]."</td></tr></table>";
$text1.= "</div>";
// цены для товара по акции
if($row1[new_pay_rub]>0)
```

```
{  
$text1.=<div class='no_pay_tovar'>  
Цена - ".$row1[pay_rub]." руб.</div>";  
$text1.=<div class='new_pay_tovar'>  
Спец. цена - ".$row1[new_pay_rub]." руб.</div>";  
}  
// цены для товара без акции  
else  
{$text1.=<div class='pay_tovar'>  
Цена - ".$row1[pay_rub]." руб.</div>";  
// скидка  
$query2="SELECT discount FROM users WHERE id='".$SESSION[user]."';  
$text1.=<div class='discount_tovar'>Скидка - ".  
mysql_result(mysql_query($query2),0)." %</div>";  
// ссылки в Корзину и Товар Подробно  
$text1.=<div>  
    <div class='function_tovar'>  
        <a href='javascript:void()'  
        onclick='xajax_Add_To_Korzina(".$row1[id].");'>  
        В корзину</a>  
        (или drag&drop img или названия)</div>  
    <div class='function_tovar'>  
        <a href='javascript:void()'  
        onclick='xajax_View_Tovar(".$row1[id].");'>Подробно</a>  
    </div>";  
// ссылка для админа Редактировать товар  
if($_SESSION[type]>7)  
{$text1.=<div class='function_tovar'>  
    <a href='javascript:void()' onclick='  
    xajax_Edit_Tovar(".$row1[id].");'>Редактировать</a>  
    </div>";  
$text1.=<div class='linedown_tovar'><hr></hr></div>";  
$text1.=</div>";  
}  
// ссылка для админа Добавить новый товар  
if($_SESSION[type]>7)  
{$text1.=<div class='function_tovar'>  
    <a href='javascript:void()' onclick='  
    xajax_Add_New_Tovar();'>Добавить новый товар</a>  
    </div>";  
// список ссылок перехода по страницам  
$text2="";  
if($pages>1)  
{  
    if($page != 1)
```

```
 {$i=$page-1;
$text2.="
```

ЗАМЕЧАНИЕ

При формировании блока ссылок в навигаторе страниц необходимо перед переходом на хайп-функцию `View_Search_Tovars` и отправкой значений полей формы поиска предусмотреть установку значения для скрытого поля `pagesearch`.

3.5.7. Просмотр товара подробно

Функция `View_Tovar`, отвечающая за вывод товара, находится в файле `prgtovars/view_tovar.php` (листинг 3.38). Входные параметры — ID выбранного товара. Получая ID товара, выводим информацию в отдельный блок. В отличие от краткого показа товара, картинку выводим крупно. Информация о товаре такая же, как и в разд. 3.5.6, только вместо краткого описания выводится полное.

Листинг 3.38

```
<?php
function View_Tovar($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // вывести контент
    $content=f_view_tovar($Id);
    $objResponse->assign("center5","innerHTML",$content);
    // вывести заголовок
    $zagcontent=f_zag1("Товар подробно");
    $objResponse->assign("centercaption5","innerHTML",$zagcontent);
    // блок center5 в зону видимости
    $objResponse->script("document.getElementById(
        'center5').scrollIntoView();");

    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Вывод результата поиска товаров иллюстрирует рис. 3.18.

ЗАМЕЧАНИЕ

Когда товар удален или спрятан, пользователь может посмотреть его только в том случае, если ранее заказывал (при просмотре своего заказа). При этом ссылка **В корзину** для удаленного или скрытого товара будет отсутствовать.

Функция `f_view_tovar`, формирующая контент, находится в файле `prgtovars/function_view_tovar.php` (листинг 3.39).

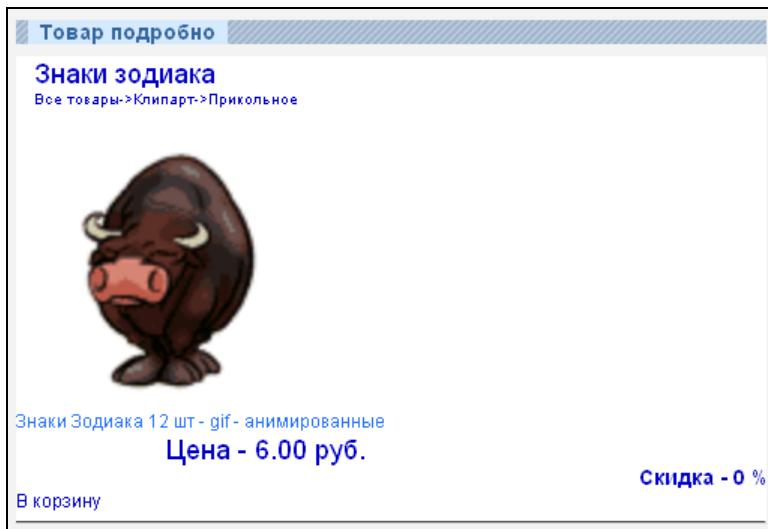


Рис. 3.18. Вывод подробной информации о товаре

Листинг 3.39

```
<?php
// Просмотр товара
// $Id - id товара
function f_view_tovar($Id)
{
    // файл настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    // подключение файла с подпрограммой формирования пути к текущей
    // категории
    require_once("prgtovars/function_string_kategory.php");
    // получение информации о товаре
    $query1="SELECT * FROM tovars WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    // наименование
    $text1."<div class='zag_tovar'>".$row1[name]."</div>";
    // путь к категории товара
    $text1."<div class='kategory_tovar'>
        ".f_string_kategory($row1[id_kategory])."</div>";
    // если картинка непустая - выводим картинку
    if($row1[img]!='imgtovar/nofoto.gif')
        $text1."<div><img src='".$row1[img]."' width=200px></div>";
```

```
// полная информация по товару
$text1.=<div class='fullinfo_tovar'>".$row1[fullinfo]."</div>";
// цены по товару по акции
if($row1[new_pay_rub]>0)
{
    $text1.= "<div class='no_pay_tovar'>Цена - ".$row1[pay_rub]." руб.</div>";
    $text1.= "<div class='new_pay_tovar'>Спец. цена - ".
        ".$row1[new_pay_rub]." руб.</div>";
}
// цены по товару без акции
else
    $text1.= "<div class='pay_tovar'>Цена - ".$row1[pay_rub]." руб.</div>";
// скидки
$query2="SELECT discount FROM users WHERE id='".$_SESSION[user]."' ";
$text1.= "<div class='discount_tovar'>Скидка - ".
mysql_result(mysql_query($query2),0)." %</div>";

$text1.= "<div>";
// ссылка в Корзину
if($row1[visible]=='yes')
    $text1.= "<div class='function_tovar'>
        <a href='javascript:void()' onclick='
            xajax_Add_To_Korzina(\".$row1[id].\");'>В корзину</a>
    </div>";
if($_SESSION[type]>7)
{
    // ссылка для админа Редактировать товар
$text1.= "<div class='function_tovar'>
    <a href='javascript:void()' onclick='
        xajax_Edit_Tovar(\".$row1[id].\");'>Редактировать</a>
    </div>";
    if($row1[visible]=='yes')
        // ссылка для админа Скрыть товар
        $text1.= "<div class='function_tovar'>
            <a href='javascript:void()' onclick='
                xajax_DoHidden_Tovar(\".$row1[id].\");'>Скрыть</a>
            </div>";
    // ссылка для админа Открыть товар для показа
    else
        $text1.= "<div class='function_tovar'>
            <a href='javascript:void()' onclick='
                xajax_DoVisible_Tovar(\".$row1[id].\");'>Открыть</a>
            </div>";
    // ссылка для админа Удалить товар
```

```

if($row1[visible]!='del')
    $text1.= "<div class='function_tovar'>
        <a href='javascript:void()' onclick='
            xajax_Delete_Tovar(\".$row1[id].\");
        '>Удалить</a>
    </div>";
}
$text1.= "<div class='linedown_tovar'><hr></hr></div>";
$text1.= "</div>";
return $text1;
}
?>

```

3.5.8. Специальные акции (товары по акции)

Реализуем еще вывод блока показа товаров по акции (специальную цену на товар устанавливает администратор). Для каждого товара кроме обычной цены можно назначить и специальную при проведении различных акций. При этом старая цена товара будет перечеркнута красным. Список товаров по акции будем выводить под заголовком "Акция" в блок с id=left2 (рис. 3.19).

Рис. 3.19. Список товаров по акции

Функция `f_view_tovar`, формирующая контент, находится в файле `prgtovars/function_view_tovars_action.php` (листинг 3.40).

Листинг 3.40

```

<?php
// Просмотр товара
// по акции
function f_view_tovars_action()
{

```

```
require_once("my.php");
require_once("mybaza.php");
$text1="";
$query1="SELECT * FROM tovars WHERE new_pay_rub>0 && visible='yes'
        ORDER BY data DESC LIMIT 0, 5 ";
$rez1=mysql_query($query1);
if(mysql_num_rows($rez1)>0)
{
    while($row1=mysql_fetch_assoc($rez1))
    {
        $text1.= "<div class='zag_tovar_action'><span
            onmousedown='var x=document.getElementById(\"windowdrag\");
            x.style.visibility=\"visible\";
            var y=\"$".$row1[name]."\";
            x.innerHTML=y;
            flag1.status=1;flag1.id=\"$row1[id]\";
            return false;'>".$row1[name]."</span></div>";
        $text1.= "<div><span class='no_pay_tovar1'>
            ".$row1[pay_rub]."</span>";
        $text1.= "<span class='new_pay_tovar1'> - ".$row1[new_pay_rub]."
            руб.</span></div>";
        $text1.= "<div>
            <div class='function_tovar'>
                <a href='javascript:void()' onclick='
                    xajax_Add_To_Korzina(\".$row1[id].\");
                '>В корзину</a>
            </div>
            <div class='function_tovar'>
                <a href='javascript:void()' onclick='
                    xajax_View_Tovar(\".$row1[id].\");
                '>Подробно</a>
            </div>";
        if($_SESSION[type]>7)
            $text1.= "<div class='function_tovar'>
                <a href='javascript:void()' onclick='
                    xajax_Edit_Tovar(\".$row1[id].\");
                '>Редактировать</a>
            </div>";
        $text1.= "<div class='linedown_tovar'><hr></hr></div>";
    }
}
else
{
    $text1=<br><center>Нет товаров по акции</center><br>";
}
return $text1;
}
?>
```

Из списка товар можно будет просмотреть подробно, отправить в корзину, редактировать (для администратора).

Этим, пожалуй, и ограничим информацию о блоке товаров и перейдем к рассмотрению корзины.

3.6. Корзина

Корзина предназначена для отбора понравившихся товаров. Хранить данные корзины можно в файлах cookies или в переменных `$_SESSION`. У нас эта информация будет храниться в текстовых файлах `f-$_SESSION[session].txt`. Функция `session_id()` возвращает текущий идентификатор сессии (SID). Для хранения файлов отводим папку `tmp1`. Сохраняться содержимое файлов будет до формирования заказа, т. е. корзина при выходе и входе с сайта не будет обнуляться, и можно ее редактировать несколько дней. Уничтожаться файл будет только при формировании заказа.

3.6.1. Добавление товаров в корзину

При входе в профиль вызывается функция `f_korzina_right`. Проверяется существование файла `f-$_SESSION[session].txt`. Если файла не существует, создаем его (пустой), иначе считываем содержимое. Функция `f_korzina_right` находится в файле `prgkorzina/function_korzina_right.php` (листинг 3.41).

Листинг 3.41

```
<?php
// Изображение корзины
// и кол-во товара в корзине
function f_korzina_right()
{
    // подключаемся к базе данных
    require_once("mybaza.php");
    $text1="";
    // проверка существования корзины
    $file1="tmp1/f-".$_SESSION[session]."-".$_SESSION[user].".txt";
    // если нет - создать
    if(!file_exists($file1))
    {
        $fp=fopen($file1,"w");chmod($file1,0777);
        $count=0;$kol=0;$summa=0;
    }
    // если есть, посчитать сумму и кол-во
    else
    {
        $fp=fopen($file1,"r");
    }
}
```

```
$count=0;$summa=0;
while($str=fgetcsv($fp,1000,";"))
{
$count++;
$kol+=$str[1];
$summa+=$str[1]*$str[2];
}
}

$text1.= "<center><img id='imgkorzina'
          onmouseover='flag1.over=1;' onmouseout='flag1.over=0;'
          src='img/korzina.jpg'><br>";
if($count>0) // корзина непустая
{
$text1.= "<br>Товаров - ".$count;
$text1.= "<br>Кол-во - ".$kol;
$text1.= "<br>".$summa." руб";
$query1="SELECT usd,eur FROM rate ORDER BY data DESC LIMIT 0, 1 ";
$rez1=mysql_query($query1);
$row1=mysql_fetch_assoc($rez1);
$usd=$row1[usd];$eur=$row1[eur];
$text1.= "<br>".sprintf("%8.2f",$summa/$usd)." usd";
$text1.= "<br>".sprintf("%8.2f",$summa/$eur)." eur";
$text1.= "<br><a href='javascript:void();' onclick='xajax_View_Korzina();'>Подробно</a>";
}
else // корзина пустая
{
$text1.= "<br>Корзина пустая";
}
$text1.= "<center>";
return $text1;
?>
```

Корзина кратко показывает:

- количество товаров;
- число наименований товаров;
- сумму в трех валютах.

Вид корзины иллюстрируют рис. 3.20 и 3.21.

В файл корзины информация о товарах записывается по строкам: ID товара; количество; цена товара.

При нажатии ссылки **В корзину** или при перетаскивании картинки (или заголовка товара) в корзину происходит вызов хайп-функции `Add_To_Korzina` (из файла `prgkorzina /add_to_korzina.php`, листинг 3.42). В качестве аргумента передается ID товара. Далее проверяем файл корзины на наличие в нем такого товара. Проверку

осуществляет функция `a_add_to_korzina`, находящаяся в файле `prgkorzina/function_add_to_korzina.php` (листинг 3.43). Если такой товар существует — изменяется количество для данного товара в файле корзины. Если такой товар отсутствует в корзине, создается новая строка в файле корзины.

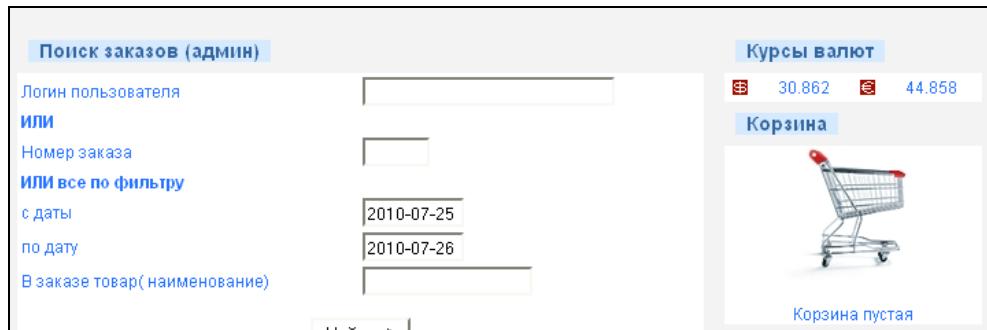


Рис. 3.20. Вид пустой корзины

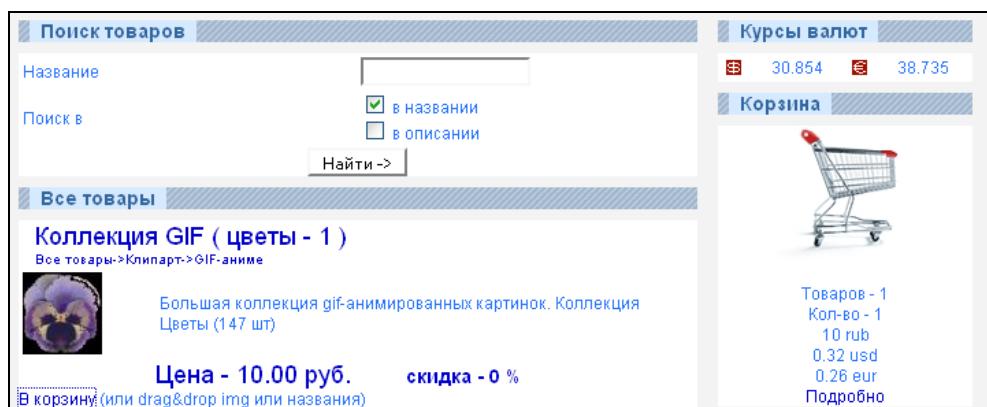


Рис. 3.21. Вид непустой корзины

Листинг 3.42

```
<?php
// Добавить товар в корзину
function Add_To_Korzina($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value", 'yes');
    // добавить товар в корзину
    $content=f_add_to_korzina($Id);
```

```
if($content=='no')
{
    // получить контент для корзины
    $content1=f_korzina_right();
    // запустить javascript - если корзина подробно видна
    // изменить ее содержимое
    $script2="if(document.forms.Flags.flag_korzina.value=='yes') ";
    $script2.="{xajax_View_Korzina();}";
    $objResponse->script($script2);
    $objResponse->assign("right2","innerHTML",$content1);
    // перенести видимость на блок корзины
    $objResponse->script("document.getElementById
        ('right2').scrollIntoView();");
}
else
{
    $objResponse->alert("Такой товар уже есть в корзине !!!");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Листинг 3.43

```
<?php
function f_add_to_korzina($Id)
{
    $text1="no";
    // подключаемся к базе данных
    require_once("mybaza.php");
    // открываем файл корзины
    $file1="tmp1/f-".$_SESSION[session]."-".$_SESSION[user].".txt";
    $fp=fopen($file1,"r");
    $tostore='';$kol=1;
    // считываем данные
    while($str=fgetcsv($fp,1000,";"))
    {
        $tostore=implode($str,";")."\r\n";
        // есть такой товар
        if($str[0]==$Id)
        {
            $text1='yes';
            $kol=$str[1]+1;
        }
        // нет такого товара
        // запоминаем в новый контент строки
```

```

else
    $tofile.= $tofile1;
}
//if($text1=='no')
//{
$query1="SELECT pay_rub,new_pay_rub FROM tovars WHERE id='".$Id."'";
$query2="SELECT discount FROM users WHERE id='".$SESSION[user]."'";
$discount=mysql_result(mysql_query($query2),0);
if(mysql_result(mysql_query($query1),0,"new_pay_rub")>0)
    $pay_rub=mysql_result(mysql_query($query1),0,"new_pay_rub");
else
    $pay_rub=mysql_result(mysql_query($query1),0,"pay_rub");
// цена сразу считается по скидке
$pay_rub=trim(sprintf("%10.2f",$pay_rub*(100-$discount)/100));
// добавляем в новый контент строки
$tofile.=$Id.";" . $kol.";" . $pay_rub . "\r\n";
//}
fclose($fp);
// запись новых данных в файл корзины
$fp=fopen($file1,"w");
fwrite($fp,$tofile);
fclose($fp);
return $text1;
}
?>

```

ЗАМЕЧАНИЕ

Задокументированные строки в листинге 3.43 — это вариант, когда при помещении такого же товара к корзину, он не добавляется, а просто выводится сообщение **Такой товар уже есть в корзине** (рис. 3.22).

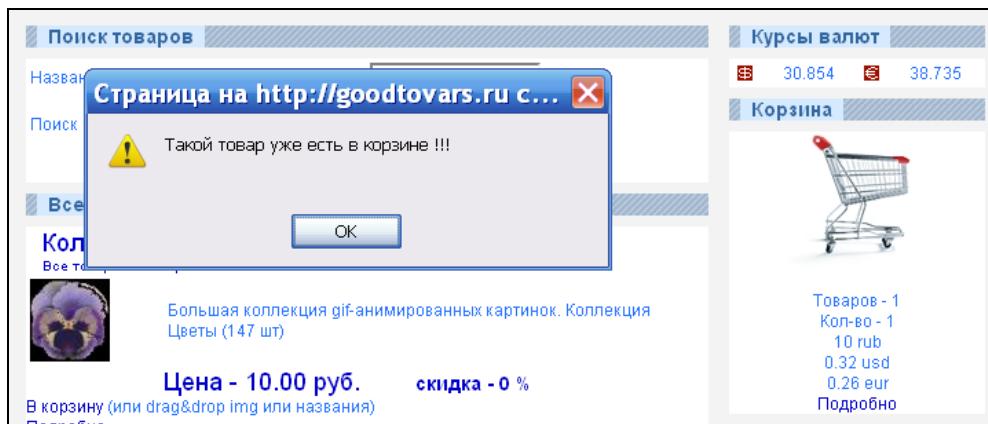


Рис. 3.22. Вид корзины

3.6.2. Корзина подробно

Блок **Корзина подробно** служит для подробного просмотра корзины. Здесь мы можем просмотреть список товаров, общую сумму, отредактировать количество, удалить товар из корзины, а также оформить заказ. Блок **Корзина подробно** открывается при нажатии ссылки **Подробно** в блоке **Корзина** (рис. 3.23). При этом происходит вызов хаях-функции `View_Korzina` (из файла `prgkorzina /view_korzina.php`, листинг 3.44).

Товар	Кол-во	Цена, руб	Сумма, руб
1 Коллекция GIF (цветы - 1)	1	10.00	10
2 Коллекция GIF (птицы - 1)	3	10.00	30
3 Телефонные коды городов России	1	10.00	10

Итого - товаров - 5 , на сумму 50 руб.

[Оформить заказ ->](#)

Помощь

Название

Поиск в в названии в описании

Курсы валют

USD 30.854 EUR 38.735

Корзина

Товаров - 3
Кол-во - 5
50 руб
1.62 usd
1.29 eur
[Подробно](#)

Рис. 3.23. Вид корзины подробно

Листинг 3.44

```
<?php
function View_Korzina()
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value", 'yes');
    // формирование контента
    $content2=f_view_korzina();
    $zagcontent2="Корзина подробно<a href='javascript:void();' onclick=''";
    $zagcontent2.= "document.getElementById(\"flag_korzina\").value=\\"no\\\";";
    $zagcontent2.= "document.getElementById(\"center1\").innerHTML=";
    $zagcontent2.= "\"><table></table>\";";
    $zagcontent2.= "document.getElementById(\"centercaption1\").innerHTML=";
    $zagcontent2.= "\"><table></table>\";'>убрать</a>";
    $zagcontent2=f_zag1($zagcontent2);
    $objResponse->assign("centercaption1","innerHTML",$zagcontent2);
    $objResponse->assign("center1","innerHTML",$content2);
    // установка флага видимости корзины подробно
```

```

$objResponse->assign("flag_korzina","value",'yes');
// перенести видимость на блок корзины
$objResponse->script("document.getElementById
    ('centercaption1').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

Формирование контента для подробного просмотра корзины осуществляется функция `f_view_korzina`, расположенная в файле `prgkorzina/function_view_korzina.php` (листинг 3.45).

Листинг 3.45

```

<?php
function f_view_korzina()
{
require_once ("mybaza.php");
$text1="";
$file1="tmp1/f-".$_SESSION[session]."-".$_SESSION[user].".txt";
$fp=fopen($file1,"r");
$text1.= "<form id='FormKorzina' action='javascript:void(null);'
    onsubmit='xajax.\\"ButtonFormKorzhina\\").disabled=true;
    xajax.\\"ButtonFormKorzhina\\").value=
    \\\"Подождите...\\";
    xajax_Create_Zakaz(xajax.getFormValues(
        \\\"FormKorzhina\\"));'>";
$text1.= "<table width=100%>";
$text1.= "<tr><td class='str0'></td>";
$text1.= "<td class='str0'>Товары</td>";
$text1.= "<td class='str0' align=right>Кол-во</td>";
$text1.= "<td class='str0' align=right>Цена,<br>руб</td>";
$text1.= "<td class='str0' align=right>Сумма,<br>руб</td>";
$text1.= "<td class='str0'></td>";
$text1.= "</tr>";
$i=0;$summa=0;$count=0;
while($str=fgetcsv($fp,1000,";"))
{
$i++;
$text1.= "<tr>";
$text1.= "<td class='str".($i%2+1)."' width=5%>".$i."
    <input type=hidden name=korzhina_id id=korzhina_id
    value='".$str[0]."' >
    </td>";
$query1="SELECT name FROM tovars WHERE id='".$str[0]."' ";

```

```
$text1.=<td class="str".($i%2+1)."' width=60%>
    ".mysql_result(mysql_query($query1),0)."</td>";
$text1.=<td class="str".($i%2+1)."' width=10%><input type=text
    name=korzina_kol".$str[0]." id=korzina_kol".$str[0]."
    value='".$str[1]."' size=3 maxlength=3 ".READONLYK."
    onchange='var x=new Array();x[0]='".$str[0]."';x[1]=this.value;
    xajax_Change_Kol_Korzina(x);'>
</td>;
$text1.=<td class="str".($i%2+1)."' width=10%><input type=text
    name=korzina_pay".$str[0]."
    id=korzina_pay".$str[0]." value='".$str[2]."' size=3
    maxlength=6 readonly
    onclick='document.getElementById(
        \'korzina_kol\'.$str[0].\').focus();
    return false;'></td>;
$text1.=<td class="str".($i%2+1)."' width=10%><input
    type=text name=korzina_summa".$str[0]."
    id=korzina_summa".$str[0]." value='".$str[2]*$str[1]."'
    size=6 maxlength=6 readonly
    onclick='document.getElementById(
        \'korzina_kol\'.$str[0].\').focus();
    return false;' ></td>;
$text1.=<td class='str'.($i%2+1)."' width=5%><a href='
    javascript:void();' onclick='
    xajax_Delete_From_Korzina(\".$str[0].\")'><img
    src='img/delete.png'>
</a></td>;
$text1.= "</tr>";
$summa+=$str[1]*$str[2];
$count+=$str[1];
}
$text1.= "</table>";
$text1.= "<br>
    <div id='itogo_korzina'>Итого - товаров - ".$count." ,
    на сумму ".$summa." руб.</div>
    <br>";
$text1.= "<center><input type='submit' id='ButtonFormKorzina'
    value='Оформить заказ -'></center>";

$text1.= "</form>";
return $text1;
}
?>
```

ЗАМЕЧАНИЕ

Константа `READONLYK` находится в файле `my.php` и отвечает за возможность изменения количества товара.

3.6.3. Редактирование корзины

К операциям редактирования корзины относятся:

- изменение количества товара в позиции корзины;
- удаление товара из корзины.

3.6.3.1. Изменение количества товара

Если значение константы `READONLYK` в файле настроек `my.php` равно `yes`, то изменение числа позиций товара в корзине разрешено. При этом по событию `onchange` вызывается хайп-функция `Change_Kol_Korzina`, расположенная в файле `prgkorzina/change_kol_korzina.php` (листинг 3.46).

Листинг 3.46

```
<?php
// Изменение корзины при изменении количества
function Change_Kol_Korzina($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // изменение количества
    $content=f_change_kol_korzina($Id);
    if($content=='yes') // успешно
    {
        // получить контент блока "корзина кратко"
        $content1=f_korzina_right();
        // получить контент блока корзина полная
        $content2=f_view_korzina();
        // отправка на выполнение javascript изменения
        // вида - корзина
        $script2="if(document.forms.Flags.flag_korzina.value=='yes')";
        $script2.="{xajax_View_Korzina();}";
        $objResponse->script($script2);
        // вывод измененного контента в блок "корзина кратко"
        $objResponse->assign("right2","innerHTML",$content1);
        // перенести видимость на блок корзины
        $objResponse->script("document.getElementById('right2').
            scrollIntoView();");
    }
    else // ошибка
}
```

```
$objResponse->alert("Ошибка изменения корзины !!!");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

Изменение файла корзины и формирование нового контента осуществляется функция `f_change_kol_korzina`, расположенная в файле `prgkorzina/function_change_kol_korzina.php` (листинг 3.47).

Листинг 3.47

```
<?php
function f_change_kol_korzina($Id)
{
    $text1="yes";
    // открытие файла корзины
    $file1="tmp1/f-".$_SESSION[session]."-".$_SESSION[user].".txt";
    $fp=fopen($file1,"r");
    $tofile='';
    while($str=fgetcsv($fp,1000,";"))
    {
        // поиск строки изменения
        // накопление строк для файла корзины
        if($str[0]==$Id[0])
        {
            $tofile.=$Id[0].";".$Id[1].";".$str[2].";"."\\r\\n";
        }
    }
    else
        $tofile.=implode($str,";")."\r\n";
    }
    fclose($fp);
    // сохранение нового содержимого
    $fp=fopen($file1,"w");
    if(!fwrite($fp,$tofile))
        $text1="no";
    fclose($fp);
    return $text1;
}
?>
```

3.6.3.2. Удаление товара из корзины

Для удаления позиции товара из корзины, необходимо нажать на иконку удаления, расположенную в строке каждой позиции товара справа (см. рис. 3.23). При удалении товара вызывается хайп-функция `Delete_From_Korzina`, расположенная

в файле prgkorzina/delete_from_korzina.php (листинг 3.48). При этом позиция удаляется из корзины (рис. 3.24), а также изменяется и содержимое блока **Корзина кратко** (см. рис. 3.23).

The screenshot shows a shopping cart summary page with the following details:

- Корзина подробно убрать** (Shopping Cart detailed remove):

Товар	Кол-во	Цена, руб	Сумма, руб
1 Коллекция GIF (птицы - 1)	3	10.00	30
2 Телефонные коды городов России	1	10.00	10

 A red 'X' icon is next to each row, indicating they can be deleted. Below the table, it says "Итого - товаров - 4 , на сумму 40 руб." (Total - 4 items, for a sum of 40 rubles).
- Оформить заказ ->** (Order now ->)
- Курсы валют** (Currency exchange rates):

руб	30.854	евро	38.735
-----	--------	------	--------
- Корзина** (Cart):

--
- Поиск товаров** (Search products):

Название	<input type="text"/>
Поиск в	<input checked="" type="checkbox"/> в названии <input type="checkbox"/> в описании
<input type="button" value="Найти ->"/>	
- Товаров - 2** (Items - 2)
Кол-во - 4 (Quantity - 4)
40 руб (40 rubles)
1.30 usd (1.30 usd)
1.03 eur (1.03 eur)
[Подробнее](#) (More details)

Рис. 3.24. Вид вкладки Корзина подробно после удаления товара

Листинг 3.48

```
<?php
// Удалить товар из корзины
function Delete_From_Korzina($Id)
{
  $objResponse = new xajaxResponse();
  $objResponse->assign("flag_ajax","value",'yes');
  // произвести удаление из файла корзины
  $content=f_delete_from_korzina($Id);
  if($content=='yes') // успешно
  {
    // получить контент блока корзина краткая
    $content1=f_korzina_right();
    // получить контент блока корзина полная
    $content2=f_view_korzina();
    // отправка на выполнение javascript изменения
    // вида - корзина
    $script2="if(document.forms.Flags.flag_korzina.value=='yes')";
    $script2.="{xajax_View_Korzina();}";
    $objResponse->script($script2);
    // вывод измененного контента в блок "корзина кратко"
    $objResponse->assign("right2","innerHTML",$content1);
    // перенести видимость на блок корзины
```

```
$objResponse->script("document.getElementById
    ('right2').scrollIntoView();");
}

else // ошибка
    $objResponse->alert("Ошибка удаления товара из корзины!!!");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}

?>
```

Изменяет файл корзины и формирует новый контент функция `f_delete_from_korzina`, расположенная в файле `prgkorzina/function_delete_from_korzina.php` (листинг 3.49).

Листинг 3.49

```
<?php
// Удаление товара из корзины
function f_delete_from_korzina($Id)
{
    $text1="yes";
    // открытие файла корзины
    $file1="tmp1/f-".$_SESSION[session]."-".$_SESSION[user].".txt";
    $fp=fopen($file1,"r");
    $tostore='';
    while($str=fgetcsv($fp,1000,";"))
    {
        // поиск строки изменения
        if($str[0]==$Id)
        ;
        // накопление строк для файла корзины
        else
            $tostore.=implode($str,";")."\r\n";
    }
    fclose($fp);
    // сохранение нового содержимого
    $fp=fopen($file1,"w");
    fwrite($fp,$tostore);
    fclose($fp);

    return $text1;
}
?>
```

3.6.4. Оформление заказа

При нажатии на кнопку **Оформить заказ** (см. рис. 3.24), данные заказа передаются хайп-функции `Create_Zakaz`, которая расположена в файле `prgzakaz/create_zakaz.php` (листинг 3.50).

Листинг 3.50

```
<?php
function Create_Zakaz($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // создание заказа, обнуление корзины
    // получение контента
    $content2=f_create_zakaz($Id);
    $zagcontent2="Создание заказа <a href='javascript:void();' onclick=''";
    $zagcontent2.="document.getElementById(\"flag_korzina\").value=";
    $zagcontent2.="\\"no\\\";";
    $zagcontent2.="document.getElementById(\"center1\").innerHTML=\\\"\\\"";
    $zagcontent2.="document.getElementById(\"centercaption1\").innerHTML=";
    $zagcontent2.="\\"\\\"; 'урвать</a>";
    // заголовок
    $zagcontent2=f_zag1($zagcontent2);
    $objResponse->script($script2);
    $objResponse->assign("centercaption1","innerHTML",$zagcontent2);
    $objResponse->assign("center1","innerHTML",$content2);
    $objResponse->assign("flag_korzina","value",'no');
    $objResponse->script("document.getElementById
        ('centercaption1').scrollIntoView();");
    // вывод контента корзины (пустая)
    $content3=f_korzina_right();
    $objResponse->assign("right2","innerHTML",$content3);
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Создает заказ, обнуляет файл корзины и формирует контент функция `f_create_zakaz`, расположенная в файле `prgzakaz/function_create_zakaz.php`. Данные для записи в таблицы берутся из файла корзины. Сначала создаем запись в таблицу `zakaz` (`id_user` берем из переменной `$_SESSION`) и получаем ID заказа (`$id=mysql_insert_id();`). Построчно считывая функцией `fgetcsv` данные из файла корзины, заносим их в таблицу `zakaz_table`. При этих операциях, подсчитываем сумму заказа и заносим в таблицу `zakaz`. Затем файл корзины очищается. Кроме

этого нам необходимо сделать записи для мгновенных оповещений на сайте для пользователя и администратора о созданном заказе. Содержимое файла prgzakaz/function_create_zakaz.php приведено в листинге 3.51.

Листинг 3.51

```
<?php
function f_create_zakaz($Id)
{
    // подключение к базе данных
    require_once ("mybaza.php");
    $text1="";$text01="";
    // извлечение из корзины и запись в базу
    $file1="tmp1/f-".$_SESSION[session]."-".$_SESSION[user].".txt";
    // создание заказа
    $query1="INSERT INTO zakaz SET id_user=''. $_SESSION[user].'";
    $rez1=mysql_query($query1);
    if (!$rez1)
        $text01.=$query1;
    // получение id нового заказа
    $id=mysql_insert_id();
    // вставка позиций в zakaz_table и вычисление суммы заказа
    $fp=fopen($file1,"r");
    $summa=0;
    while ($str=fgetcsv($fp,1000,";"))
    {
        $id_zakaz=$id;
        $id_tovar=$str[0];
        $kol=$str[1];
        $pay_rub=$str[2];
        $summa_rub=$str[1]*$str[2];
        $summa_rub_oplata=0;
        $link="0";
        $count_download=0;
        $query2="INSERT INTO zakaz_table SET
            id_zakaz='".$id_zakaz."',id_tovar='".$id_tovar."',
            pay_rub='".$pay_rub."',kol='".$kol."',
            summa_rub='".$summa_rub."',id_link='".$link."',
            count_download='".$count_download."'";
        $rez2=mysql_query($query2);
        $summa+=$str[1]*$str[2];
        $query02="SELECT name FROM tovars WHERE id='".$str[0]."';
        $text01.=mysql_result(mysql_query($query02),0)." - ".$str[1]."<br>";
```

```

}

// запись в таблицу zakaz
$id_user=$_SESSION[user];
$summa_rub=$summa;
$summa_rub_oplata=0;
$visible='yes';

$query3="UPDATE zakaz SET data='".date('Y-m-d H:i:s')."',
    id_user='".$id_user."',summa_rub='".$summa_rub."',
    pay='no',visible='".$visible."'";
    WHERE id='".$id."' ";

$rez3=mysql_query($query3);

// message_header4
f_create_message_header4(5,$id, $_SESSION[user]," ".date('Y-m-d
H:i:s')." создан заказ ".$id." на сумму ".$summa_rub." руб. ");

fclose($fp);

// очистка файла корзины
$fp=fopen($file1,"w");
fwrite($fp,"");
fclose($fp);

// формирование страницы вывода
$text1."<center>";
$text1."<br><br><b>Заказ ".$id."</b><br><br>";
$text1.=$text01;
$text1.="  
 Сумма к оплате : <br><b>".$summa." руб. </b> или <br>";
$query4="SELECT usd,eur FROM rate ORDER BY data DESC LIMIT 0, 1 ";
$rez4=mysql_query($query4);
$row4=mysql_fetch_assoc($rez4);
$usd=$row4[usd];$eur=$row4[eur];
$text1."<b>".sprintf("%8.2f",$summa/$usd)." usd </b> или";
$text1."<br><b>".sprintf("%8.2f",$summa/$eur)." eur </b>";
$text1."<br><br><input type=button value='Оплатить' onclick='
    xajax_Oplata_Zakaz(\".$id.\");' >";
$text1."<br><br><input type=button value='Отложить' onclick='
    document.getElementById(\"center1\").innerHTML=\"";
    document.getElementById(\"centercaption1\").innerHTML=
    \"\";\'>";
$text1."</center>";

return $text1;
}

?>

```

Далее система предлагает либо оплатить заказ сразу, либо отложить это действие (рис. 3.25).

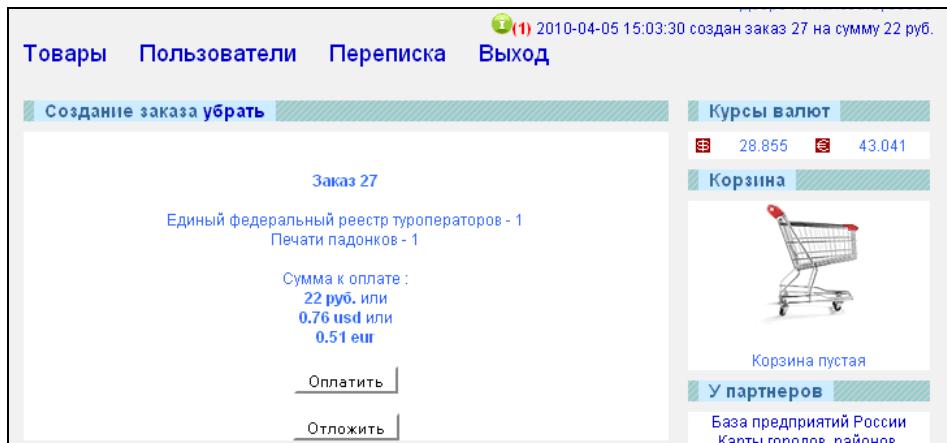


Рис. 3.25. Создание заказа. Переход на оплату

ЗАМЕЧАНИЕ

При создании заказа сообщение для блока мгновенных сообщений (см. рис. 3.25) формируется для текущего пользователя и администратора.

3.7. Оплата заказа

После создания заказа, система предлагает оплатить его (см. рис. 3.25). При нажатии на кнопку **Оплатить** предлагается выбор — автоматическая оплата через сервис либо WM (**Merchant.webmoney.ru**), либо ONPAY(<http://onpay.ru>) (рис. 3.26).

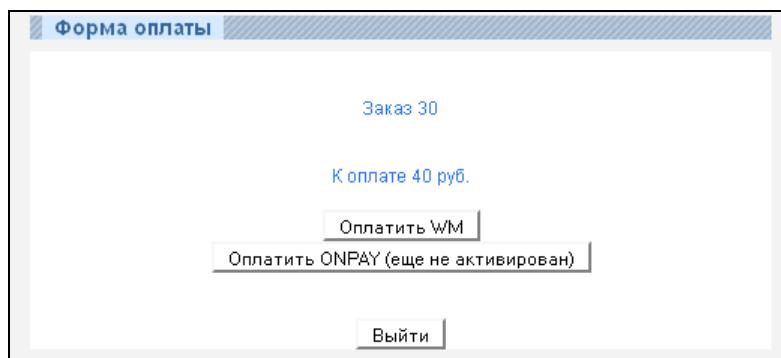


Рис. 3.26. Выбор формы оплаты

3.7.1. Оплата Webmoney

Функция `Oplata_Zakaz`, расположенная в файле `prgoplata/ oplata_zakaz.php` (листинг 3.52), создает форму выбора оплаты.

Листинг 3.52

```
<?php
//Функция округления для md5
function to_float($sum)
{
    if (strpos($sum, "."))
        {$sum=round($sum,2);}
    else
        {$sum=$sum.".0";}
    return $sum;
}
// Оплата заказа
function Oplata_Zakaz($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // очистить блок centercaption1
    $objResponse->assign(
        "centercaption1","innerHTML","<table></table>");
    // очистить блок centercaption1
    $objResponse->assign("center1","innerHTML","<table></table>");
    require_once("my.php");
    // подключиться к базе данных
    require_once("mybaza.php");
    $query1="SELECT * FROM zakaz WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    $text1."<center>";
    $text1.="<br><br>Заказ ".$row1[id]."<br>";
    $text1.="<br><br>К оплате ".$row1[summa_rub]-
        $row1[summa_rub_oplata]." руб.<br><br>";
    // WebMoney
    $text1.="

```

```

    onclick='this.disabled=true;this.value=\"Подождите...\";'>
    </form>";
// end WebMoney
// ONPAY
$key=OP_KEY;
$code=$row1[id];
$summa=$row1[summa_rub]-$row1[summa_rub_oplata];
$sum_for_md5=to_float($summa);
// Создаем проверочную строку, которая защищает платежную ссылку от
// изменений
$md5check=md5("fix;$sum_for_md5;RUR;$code;yes;$key");
$url_onpay="http://secure.onpay.ru/pay/".OP_LOGIN."?pay_mode=
fix&pay_for=".$code;
$url_onpay.="&price=". $summa ."&currency=RUR&convert=
yes&md5=". $md5check;
$url_onpay.="&price=". $summa ."&currency=RUR&convert=yes";
// $url_onpay.="&url_success=".OP_PATH;
$text1.= "<form action='javascript:void()' onclick='
    window.open(\"".$url_onpay."\",\"\",\"\");'>
    input type='submit' id='button_oplata_wm' value='
        Оплатить ONPAY '
    disabled=true onclick='this.disabled=true;this.value=
        \"Подождите...\";'></form>";
// end ONPAY
$text1.= "<br><br><input type=button value='Выйти' onclick='
    document.getElementById(\"center5\").innerHTML=\"";
    document.getElementById(\"centercaption5\").innerHTML=\"";
    '>';
$text1.= "</center>";

$zag=f_zag1("Форма оплаты");
$objResponse->assign("centercaption5","innerHTML",$zag);
$objResponse->assign("center5","innerHTML",$text1);
$objResponse->script("document.getElementById('center5')
    .scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

При нажатии на кнопку **Оплатить WM** клиент перенаправляется на специальный сайт Мерчант-сервиса, где производит авторизацию и затем оплату со своего кошелька. При этом передаются следующие параметры:

- LMI_PAYMENT_AMOUNT — сумма платежа;
- LMI_PAYMENT_DESC — назначение платежа;

- LMI_PAYEE_PURSE — номер кошелька;
- LMI_PAYMENT — номер платежа в магазине.

Сразу же после оплаты сам сервис связывается с сервером магазина и извещает его о том, что произведена оплата такого-то заказа на такую-то сумму. Данные передаются скрипту prgoplata/wm_result.php (листинг 3.53). Скрипт магазина изменяет статус заказа в базе данных, а также формирует уведомления по e-mail и сервисы мгновенных оповещений покупателю и администратору об успешной оплате заказа.

Листинг 3.53

```
<?php
require_once("../my.php");
require_once("../mybaza.php");
require_once("function_create_message_header4.php");
// Если это форма предварительного запроса, то идем дальше...
IF($_POST['LMI_PREREQUEST']==1)
{
    // 1) Проверяем, не произошла ли подмена суммы и кошелька.
    if( trim($_POST['LMI_PAYEE_PURSE'])==LMI_PAYEE_PURSE_R )
    {
        echo "YES";
        exit;
    }
}
else
{
    echo "ERR: Неверный кошелек или сумма платежа ";
    exit;
}
// Если нет LMI_PREREQUEST, следовательно, это форма оповещения о //платеже...
ELSE
{
    // Задаем значение $secret_key.
    // Оно должно совпадать с Secret Key, указанным нами в настройках кошелька.
    $secret_key=SECRET_KEY;
    // Склеиваем строку параметров
    $common_string =
$_POST['LMI_PAYEE_PURSE'].$_POST['LMI_PAYMENT_AMOUNT'].$_POST['LMI_PAYMENT_NO'] .
$_POST['LMI_MODE'].$_POST['LMI_SYS_INVS_NO'].$_POST['LMI_SYS_TRANS_NO'] .
$_POST['LMI_SYS_TRANS_DATE'].$secret_key.$_POST['LMI_PAYER_PURSE'].$_
POST['LMI_PAYER_WM'];
    // Шифруем полученную строку в MD5 и переводим ее в верхний регистр
    $hash = strtoupper(md5($common_string));
    // Прерываем работу скрипта, если контрольные суммы не совпадают
    if($hash!=$_POST['LMI_HASH'])
```

```
exit;
else
{
$summa_rub=$_POST['LMI_PAYMENT_AMOUNT'];
$kod=$_POST['LMI_SYS_TRANS_NO'];
$id_zakaz=$_POST['LMI_PAYMENT_NO'];
$query1="UPDATE oplata SET data='".date("Y-m-d
H:i:s")."',id_zakaz='".$id_zakaz."', kod='".$kod."',
plat_system='wm',summa_rub='".$summa_rub."' ";
$rez1=mysql_query($query1);
$query1="INSERT INTO oplata SET data='".date("Y-m-d
H:i:s")."',id_zakaz='".$id_zakaz."',,
kod='".$kod."',plat_system='wm',summa_rub='".$summa_rub."' ";
$rez1=mysql_query($query1);
$query11="SELECT summa_rub_oplata,summa_rub FROM zakaz
WHERE id='".$id_zakaz."'";
$summa_rub_oplata=mysql_result
(mysql_query($query11),0,"summa_rub_oplata");
$summa_rub_zakaz=mysql_result(mysql_query($query11),0,"summa_rub");
$summa_rub_oplata_new=$summa_rub_oplata+$summa_rub;
$query12="UPDATE zakaz SET summa_rub_oplata='".$summa_rub_oplata_new."'"
WHERE id='".$id_zakaz."'";
$rez12=mysql_query($query12);
$Id=$id_zakaz;
if($summa_rub_oplata_new>=$summa_rub_zakaz)
{
$query13="UPDATE zakaz SET pay='yes' WHERE id='".$Id."'";
$rez13=mysql_query($query13);
// создание ссылок
$data=date('Y-m-d',strtotime('now+10days'));
$query2="SELECT id,id_tovar FROM zakaz_table WHERE id_zakaz='".$Id."'";
$rez2=mysql_query($query2);
while($row2=mysql_fetch_assoc($rez2))
{
$query3="SELECT arhiv FROM tovars WHERE id='".$row2[id_tovar].."'";
$link=mysql_result(mysql_query($query3),0);
$query4="SELECT id_user FROM zakaz WHERE id='".$Id."'";
$id_user=mysql_result(mysql_query($query4),0);
$query5="INSERT INTO link_downloads SET id_user='".$id_user."',
file='".$link."',status='yes',data='".$data."',id_zakaz='".$Id."'";
$rez5=mysql_query($query5);
$id_link=mysql_insert_id();
$query6="UPDATE zakaz_table SET id_link='".$id_link."'"
WHERE id='".$row2[id].'";
$rez6=mysql_query($query6);
```

```

        }
    }

$query7="SELECT id FROM users WHERE type='9' ";
$id_user1=mysql_result(mysql_query($query7),0,"id");
$query8="SELECT id_user FROM zakaz WHERE id='".$id_zakaz."'";
$id_user2=mysql_result(mysql_query($query8),0,"id_user");
// message_header4
f_create_message_header4(6,$id_zakaz, $id_user1," ".date('Y-m-d
H:i:s')." оплата по заказу ".$id_zakaz." на сумму
".$summa_rub." руб. ");
f_create_message_header4(6,$id_zakaz, $id_user2," ".date('Y-m-d
H:i:s')." оплата по заказу ".$id_zakaz." на сумму
".$summa_rub." руб. ");
}
}
?>

```

3.7.2. Организация приема платежей Webmoney

Рассмотрим настройку приема оплаты через сервис WebMoney. Для приема на сайте оплаты через сервис WebMoney необходимо иметь аттестат продавца. Рассмотрим, как настроить кошелек WM для приема оплаты на сайте. Настройка показана для Webmoney Keeper Light. Заходим в свой кошелек (рис. 3.27).

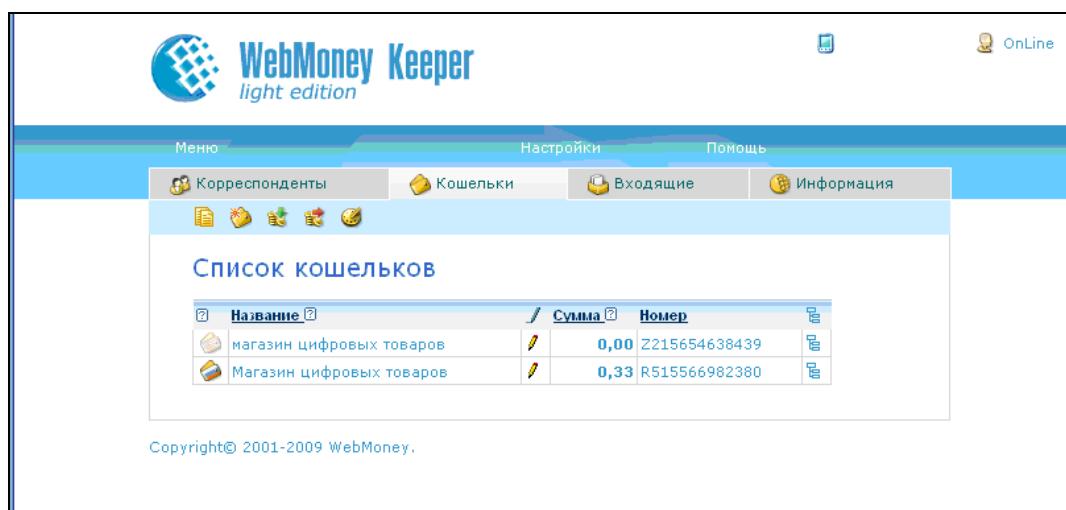


Рис. 3.27. Список кошельков

Выбираем вкладку **Информация** и нажимаем на ссылку **Как самому прода-
вать за Webmoney** (рис. 3.28).

В появившемся окне (рис. 3.29) нажимаем на ссылку **Merchant.webmoney.ru**.

Информация

Мои WebMoney

- Как [заполнить кошелек, получить, перевести, вывести WM.](#)
- Безопасность и [аттестат.](#)
- Найти другого участника системы: [где и как?](#)

WebMoney можно [заработать, получить в кредит, обменять, превратить в чеки Paymet и обратно, инвестировать в золото.](#)

А имеющиеся — потратить на покупки или оплату услуг:

Мобильная связь	Электронные книги
Платное телевидение	MP3 Музыка
Охрана и ЖКХ	Онлайн-игры
Телефония	Букмекеры и покер
Доступ в интернет	Интернет-магазины
Хостинг и домены	Гостиницы, билеты
Реклама в Сети	СМИ и периодика
Социальные сети и сообщества	Концерты, кино, шоу

[Полный каталог услуг >](#)

Как [самому продавать](#) за WebMoney.

Рис. 3.28. Вкладка Информация

Торговая система [TRUSTBAY](#) создана для того чтобы помочь продавцам и покупателям различных товаров и услуг успешно совершать сделки и своевременно получать товары или оплату. Благодаря реализованной в системе депозитной модели создаются условия, при которых нарушать заявленные обязательства становится экономически невыгодно.

Для приема оплаты за товары или услуги вы можете использовать сервис [Merchant.webmoney.ru](#). Сервис предлагает несколько интерфейсов, реализующих автоматизированную обработку платежей клиентов и проведение выплат:

- [Web Merchant Interface](#) — интерфейс автоматического приема оплаты за предоставляемые вами товары или услуги;
[настроить прием WM на сайте за три минуты](#)
- [Click&Buy Merchant Interface](#) — интерфейс, позволяющий оплачивать и получать товары на вашем сайте одним кликом мыши;
- [WM Automation Interface](#) — набор интерфейсов для создания программ, выполняющих операции в системе WebMoney Transfer с использованием протоколов HTTPS и XML.

Рис. 3.29. Выбор ссылки Merchant.Webmoney.ru

Попадаем на форму входа в защищенную зону сайта (рис. 3.30). В окне вводим код и переходим на страницу настройки кошельков (рис. 3.31). Покажем, как настроить кошелек для приема платежей на сайте. Нажимаем на ссылку **Настроить** для выбранного кошелька и попадаем на форму настройки параметров (рис. 3.32). Заполняем форму по шаблону рис. 3.32. Обратите внимание, что выбран тестовый режим, позволяющий проверить работу приема платежей на сайте без списывания денег с кошельков.

WebMoney Login

Вход в защищенную зону сайта: [Merchant WebMoney Transfer](#)
Url: <https://merchant.webmoney.ru/conf/>

Войти при помощи WebMoney Keeper Light

1. Введите код:
 указанный ниже:

80410 << если не видна картинка

2. Нажмите "Войти"
3. В появившемся диалоге выберите нужный WMID

Войти! **Отмена**

Рис. 3.30. Авторизация при входе в защищенную зону

WMID: 342813401784

[Описание сервиса](#)

Настройки

- Атtestация
- Web Merchant Interface**
- Инструкции
- Форум

[merchant.webmoney.ru / Настройки / Web Merchant Interface](#)

Список кошельков

кошелек	сумма	активность	тестовый/ рабочий	название	торговое имя	
R515566982380	0.33	Выкл.	-	Магазин цифровых товаров	Магазин цифровых товаров	настроить
Z215654638439	0.00	Выкл.	-	магазин цифровых товаров		настроить

WM-идентификатор	Параметр	активность	Описание
342813401784	Принимать платежи от неавторизованных	Вкл.	Прочитать

Рис. 3.31. Страница настройки кошельков

Рис. 3.32. Настройки торгового кошелька

После этого нажимаем на кнопку **Сохранить**. На сайте в файле настроек my.php необходимо установить значения констант

```
define(LMT_PAYEE_PURSE_R, 'R304300549042');
define(SECRET_KEY, '12345');
```

3.7.3. Платежный интегратор ONPAY

Платежный интегратор "OnPay" предлагает услугу по организации приема электронных платежей на вашем сайте всеми наиболее распространенными платежными системами Интернета. Собранные электронные деньги можно автоматически отконвертировать и вывести в другие, нужные вам электронные системы платежей или на расчетный счет. Собранная выручка исчисляется в рублях (для иностранных компаний — в долларах США или в иной валюте). Размер комиссии за прием платежей составляет всего 1%, что вместе с комиссиями самих систем оплаты в Интернете составит до 1,5–6% от суммы принятого электронного платежа. Собранные интернет-платежи вы сможете забирать сразу. Уведомление о поступлении платежа передается в он-лайн-режиме по протоколу передачи данных (API) и по электронной почте, кроме этого все оплаты через Интернет видны в реальном времени в кабинете продавца на сайте нашей системы электронных платежей. Все операции легальны на территории России и оформляются официальным договором на прием платежей, удовлетворяющим требованиям контролирующих и фискальных органов. Для подключения к сервису, вам необходимо зарегистрироваться и обязательно связаться со службой поддержки для проверки и активации аккаунта. Регистрации, не подтвержденные обращением в службу поддержки, автоматически удаляются.

3.7.3.1. Варианты приема электронных платежей

Существует два основных типа электронных платежей:

- "с конвертацией" — все поступающие в интернет-магазин платежи автоматически конвертируются в выбранную вами валюту (например, рубли к зачислению на расчетный счет, WMR, WMZ и т. д.);
- "без конвертации" — платежи остаются в том виде, в каком поступили (например, WMR, WMZ, Яндекс.Деньги и т. д.). Вы можете самостоятельно их вывесить в исходном виде или потом обменять на другую валюту.

Условно платежи можно разделить на две группы:

- "фиксированные" — это платежи на фиксированную сумму, равную стоимости товара. В зависимости от того, какая электронная платежная система выбрана, пользователю будет предлагаться к оплате различная сумма, с учетом комиссии выбранной системы. Она автоматически рассчитывается так, чтобы магазин получил сумму, точно равную стоимости товара. Обычно такой способ используется для продажи товаров магазинами;
- "свободные" — это электронные платежи на произвольную сумму, как правило, используются интернет-сервисами для пополнения баланса пользователя на сайте. В этом случае все комиссии удерживаются из суммы платежа и Мерчант получает сумму уже за вычетом комиссий платежных систем.

Платежи могут быть "по заказу" или "прямые".

При платеже "по заказу" пользователь вводит в платежную форму необходимые данные, нажимает кнопку **Далее** и перенаправляется на сайт выбранной системы электронных платежей или получает инструкцию по оплате через терминал, SMS или банковским переводом. Одновременно в системе создается заявка на платеж, которую видит Мерчант в своем кабинете. При поступлении электронного платежа заявка меняет статус на "оплачено". В платежной форме передается уникальный номер платежа, номер счета/заказа/артикула, за который производится платеж, e-mail плательщика и комментарий к платежу.

При "прямом" платеже заявка не создается, и плательщик платит на свой счет у Мерчанта в системе. Счет выглядит как XXXX******, где первые четыре цифры XXXX — это номер Мерчанта в системе OnPay, а следующие цифры ***** — номер заказа/счета/артикула на сайте у Мерчанта. "Прямые" платежи могут быть только "свободными". Для приема платежей могут использоваться такие сервисы, как "платежная форма" так и "кнопка оплатить". Примерный вид платежной формы представлен на рис. 3.33. Дизайн этой формы приема платежей можно с помощью CSS подогнать под дизайн вашего сайта. Ссылка на образец CSS лежит в кабинете продавца. "Кнопка оплатить" создается для каждого вида платежной системы отдельно, в любом дизайне. При ее нажатии пользователь сразу переходит на сайт выбранной системы с уже заданными параметрами платежа. Эта платежная форма доступна только мерчантам, подписавшим "бумажный" договор.

Опция **SMS-платежи** выбирается из общего списка в платежной форме, после чего плательщику предлагается отправить SMS с определенным текстом на короткий номер. При поступлении SMS в интернет-магазин на счет клиента мгновенно зачисляется сумма, эквивалентная WMZ. SMS-платежи принимаются почти от всех операторов сотовой связи с территории РФ и СНГ. Стоимость SMS для отправителя от 0,5 до 10 долл., Мерчант получает от 40 до 70% ее стоимости.

Шаг 1

Вы платите через платежный сервис
ON PAY RU

на сайт продавца: <http://myshop.ru/>
 его рейтинг -340 отзывы +0 -1

Отправить **Webmoney Рубли**

Сумма **1250 WMR**

Зачислить **1250 RUR**

Платеж за **555**

Ваш email **test@mail.ru**

Телефон

Я осознаю, что ответственность за качество и поставку товара несет продавец и принимаю условия [соглашения](#)

Код **8077** **8077**

Курс: **1 WMR = 1 RUR**
 Минимальный платеж: **5 WMR**

Продолжить »

Шаг 2

Оплата

Продавцу на сайте: <http://myshop.ru/>
 WMID продавца: R123456789012
 заказ/счет: **555**

за товар или услуги: Компьютеры, ноутбуки, периферия
 суммой: **1250.0 WMR**
 через систему: **Webmoney Рубли**

Внимание!
 У вас есть **15 минут** для совершения оплаты.

Нажав кнопку вы будете перемещены на сайт выбранной платежной системы, где сможете завершить платеж.

Продолжить »

Платежный интегратор Onpay.ru [аттестат](#)
WebMoney
 аттестат продавца

Рис. 3.33. Вид платежной формы

3.7.3.2. Настройка параметров магазина

После регистрации в системе ONPAY необходимо настроить магазин. Переходим на сайт <http://www.onpay.ru>. В появившемся окне (рис. 3.34) нажимаем на ссылку **Вход в систему** и попадаем на страницу входа в личный кабинет (рис. 3.35).

Вводим логин и пароль, заходим в личный кабинет, выбираем вкладку **Настройки магазина**, заполняем по образцу (рис. 3.36, 3.37) и нажимаем на кнопку **Сохранить**.

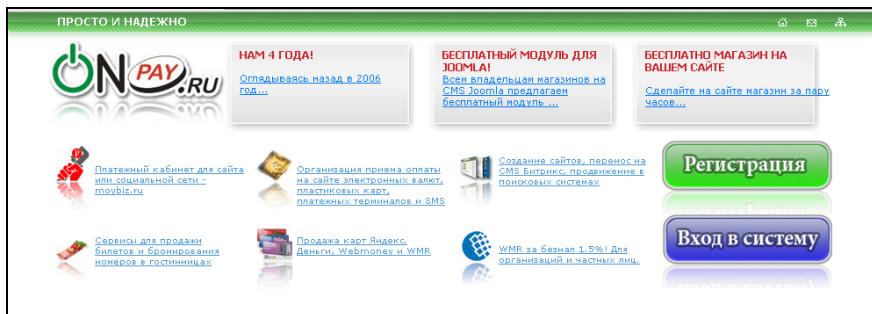
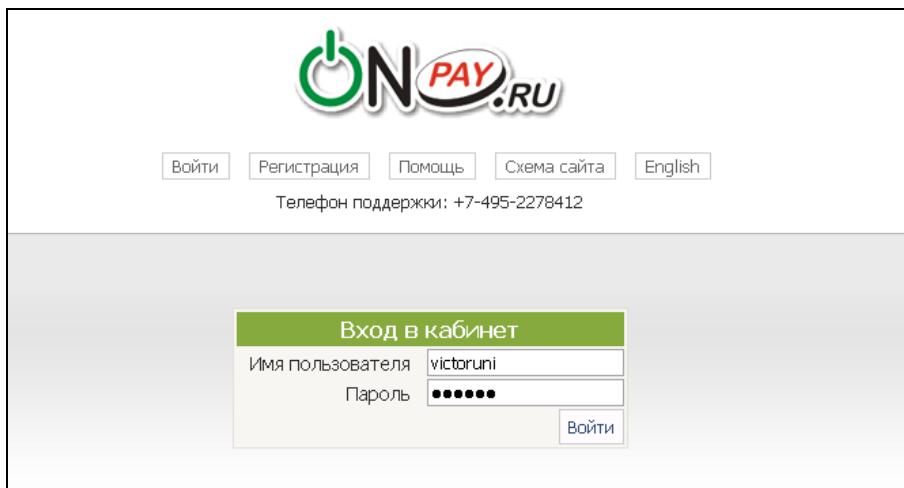
Рис. 3.34. Главная страница сайта www.onpay.ru

Рис. 3.35. Форма входа в личный кабинет

[Настройки магазина](#) [Конструктор ссылок](#) [Платежные системы](#) [Тестирование](#) [Логи](#)

Просто разместите этот код у себя на сайте вы сможете принимать платежи прямо сейчас.
<iframe src="https://secure.onpay.ru/pay/victoruni" width="300px" height="500px" frameborder=0 scrolling=no />
Внимание! У Вас должен быть активирован счет и включен хотя бы один способ приема платежей в Вашем кабинете!

Сервис

Настройки API IN	Настройки уведомлений
-------------------------	------------------------------

Адрес платежной формы: <http://secure.onpay.ru/pay/victoruni>

Прямой номер в опрau: [1233](tel:1233)

Название сервиса: Интернет-магазин

Описание товара или услуги: Продажа скриптов, справочной информации и других цифровых Товаров

Адрес сайта (для посетителей): <http://goodtovars.ru/>

уведомлять по API

Настройки магазина

В разделе "Сервис" вы размещаете описание вашего сайта, которое будет размещено в каталоге нашего сайта в разделе "Партнеры". В разделе "Настройки API IN" указываются параметры для автоматического внесения информации об оплате в информационную базу Вашего магазина.

Рис. 3.36. Вкладка Настройки магазина

Метод отправки: POST
 запросов в API: GET

URL API: <http://goodtovars.ru/prgoplata/onpc>

Пароль для API IN: 040373191066

Уведомлять о платежах на email

Email для уведомлений: На этот адрес высылаются уведомления о платежах

— Контактная информация (доступна пользователям)

Телефон:	+7 (918) 785-57-18	Контактная информация Контакты для пользователей будут видны пользователям, оплачивающим ваши товары. Обязательно должно быть указано несколько контактов для оперативного решения вопросов по зачислению платежей и получению купленных товаров или услуг.
Адрес:		
Email службы поддержки:	my_shop@bk.ru	
ICQ:	385771293	
Skype:		
Другие:		
Ссылка на форму обратной связи:		http://goodtovars.ru/index.php
Webmoney WMID:		131672524818

Рис. 3.37. Настройка параметров магазина

3.7.3.3. ONPAY Merchant API

3.7.3.3.1. Описание и очередность транзакций

Чтобы позволить системе OnPay проверить действительность ID Клиента или ID заказа, за который платит Клиент, и получить извещение о полученном платеже, вам нужно:

- активировать функцию "Уведомление по API" в разделе Мерчанта вашего профиля в OnPay;
- выставить API URL в соответствии со скриптом API на вашем сервере;
- установить секретный ключ для "уведомлений по API", который должен быть таким же, как в скрипте на вашем сервере, чтобы позволить генерацию подписей для проверок безопасности.

OnPay производит два вида запросов к API Мерчанта:

- запрос "check" используется, чтобы получить разрешение от системы Мерчанта на прием платежа от Клиента. После удачного получения разрешения, OnPay одобрит платеж. С этого момента, если Клиент действительно производит платеж, Мерчант может видеть его в разделе "мой счет" на сайте OnPay;
- запрос "pay" является, по сути, уведомлением для системы Мерчанта о том, что для него принят платеж. После получения уведомления, система Мерчанта может автоматически отправить заказанные товары или сервисы Клиенту.

Очередность транзакции:

- клиент делает платеж Мерчанту;
- OnPay отправляет "check" запрос в API Мерчанта, удостоверяясь, что система Мерчанта может (и разрешает) принять платеж;

- система Мерчанта проверяет все параметры запроса (существуют ли в системе ID Клиента и заказа, может ли Клиент платить и т. д.);
- если API Мерчанта не позволяет перевод (любой итог, кроме получения кода 0 от системы Мерчанта) — платеж не будет принят от Клиента;
- если API Мерчанта разрешает перевод (код 0) — OnPay разрешает Клиенту платить;
- клиент производит оплату, OnPay сохраняет платеж со статусом "получен" и отправляет запрос типа "pay" в API Мерчанта с теми же параметрами, что и запрос "check", плюс ID платежа и дата/время момента, когда платеж был одобрен;
- если API Мерчанта приняло уведомление (код 0), OnPay изменяет статус этого платежа с "получен" на "принят";
- если API Мерчанта сообщает о некритичной ошибке, OnPay попробует извес-тить API Мерчанта позже. OnPay будет посылать извещения с возрастающими временными интервалами, до тех пор пока API Мерчанта не примет уведомле-ние или пока не пройдет 72 часа.

Платежи со статусами "получен" и "принят" могут быть просмотрены Мерчан-том в разделе "Мой счет" на сайте OnPay. Мерчанты могут помечать платежи, как "принятые" вручную в разделе "Мой счет", если API не доступен.

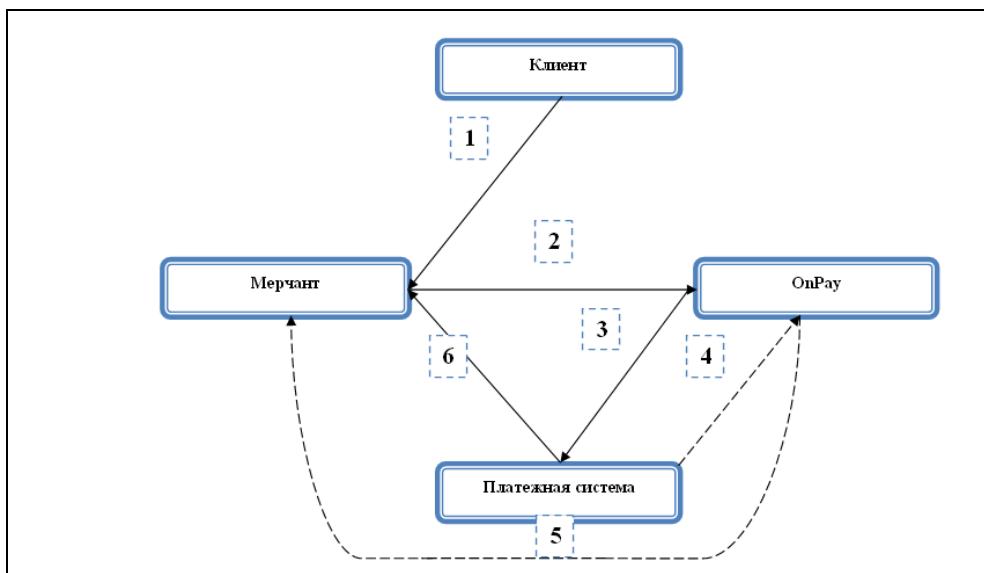


Рис. 3.38. Схема обработки платежа

Схема обработки платежа (рис. 3.38) следующая:

- клиент хочет оплатить заказ на сайте Мерчанта;
- сайт Мерчанта перенаправляет Клиента (или Клиент просто переходит по ссылке на сайте Мерчанта) на платежную страницу системы OnPay (<http://secure.onpay.ru/pay/{логин мерчанта}>);

- клиент подтверждает сумму платежа и номер заказа на платежной странице системы OnPay и переходит на сайт Платежной системы (Webmoney, Yandex Money и т. п.);
- когда Клиент успешно переводит свой платеж в Платежной системе, Платежная система автоматически уведомляет систему OnPay;
- когда OnPay получает уведомление от Платежной системы, OnPay уведомляет об этом систему Мерчанта (по API и\или отправкой сообщений на e-mail администратора Мерчанта);
- клиент может получить свой заказ от Мерчанта.

3.7.3.3.2. Параметры запросов

Параметры запросов OnPay в API Мерчанта приведены в табл. 3.4.

Таблица 3.4. Параметры запросов OnPay

Название	Возможные значения	Формат	Необходимость	Описание
onpay_id	От 1 до 32 цифр	integer	Да (присутствует только в запросах "pay")	ID платежа в системе OnPay
pay_for	От 1 до 32 символов (латинские буквы и цифры)	integer	Да	ID Клиента или заказа в системе Мерчанта, для которых производится этот платеж
order_amount	> 0	float	Да	Сумма платежа, как в атрибуте "price" платежной ссылки
order_currency	3-символьное наименование платежной системы	string	Да	Валюта, как в атрибуте "currency" платежной ссылки
balance_amount	> 0	float	Да	Сумма, которая будет внесена на баланс Мерчанта
balance_currency	3-символьное наименование платежной системы	—	Да	Валюта, в которой сумма платежа будет зачислена на баланс Мерчанта
exchange_rate	От 1 до 10 символов (латинские буквы и цифры)	float	Нет	Курс обмена, по которому сумма заказа ("order_amount") была конвертирована в сумму к получению ("balance_amount")

Таблица 3.4 (окончание)

Название	Возможные значения	Формат	Необходимость	Описание
type	check pay	string	Да	"check"-запрос — это запрос на проверку возможности оплаты указанного счета, "pay"-запрос — это уведомление о платеже, поступившем на счет Мерчанта
comment	От 0 до 255 символов (латинские буквы и цифры)	string	Опционально	Заметка, включенная в платежную форму в системе Мерчанта. Будет доступна в списке платежей в интерфейсе Мерчанта
Payment DateTime	—	dateTime	Да (присутствует только в запросах "pay")	Дата и время, в которое платеж был получен системой OnPay от Клиента
md5	40 символов (латинские буквы и цифры)	string	Да	MD5 — это хэш-подпись платежа. Стока в верхнем регистре

3.7.3.3.3. Формат даты/времени

Для определения даты OnPay использует один из форматов рекомендованных стандартом ISO8601:2000. Этот формат включен в "XML Schema Part 2: Datatypes" под именем `dateTime`:

MM-DDThh:mm:ss.fZZZZZ

Здесь:

- YYYY — год, четыре цифры;
- MM — месяц, две цифры (01 — Январь и т. д.);
- DD — день, две цифры (от 01 до 31);
- T — символ "T", верхний регистр;
- hh — hours, две цифры (24-часовой формат, от 00 до 23);
- mm — minutes, две цифры (от 00 до 59);
- ss — seconds, две цифры (от 00 до 59);
- f — от одной до шести цифр, доли секунды;
- ZZZZZ — временная зона, в формате +чч:мм или -чч:мм от UTC. Установленный символ Z означает время UTC.

Примеры:

2006-03-24T19:00:00+03:00 — 19 часов 24 марта 2006 года, зона — UTC + 3 часа.

2006-03-24T16:00:00Z — та же дата, но в зоне UTC.

Формат подписи MD5 для "check"-запросов от системы OnPay:

type;pay_for;order_amount;order_currency;secret_key_for_api_in

Формат подписи MD5 для "pay"-запросов от системы OnPay:

type;pay_for;onpay_id;order_amount;order_currency;secret_key_for_api_in

3.7.3.3.4. Поддержка различных валют

Параметры `order_amount` и `order_currency` будут теми же, что и в ссылке для заказа на сайте Мерчанта, через которую Клиент отсылается на платежную страницу системы OnPay. Из этих параметров OnPay подсчитывает сумму платежа во всех других возможных валютах.

Параметры `balance_amount` и `balance_currency` — это реальная сумма в валюте, которая поступает на баланс Мерчанта.

При проверке заказа (чтобы убедиться, что заказ реально оплачивается требуемой суммой) необходимо сверять только `order_amount` и `order_currency` со стоимостью заказа в вашей системе. Параметры `balance_amount` и `balance_currency` обычно используются для уведомлений (отчетов).

Пример:

http://secure.onpay.ru/pay/merchant_login?pay_mode=fix&price=100¤cy=USD&pay_for=12

Ссылка с суммой заказа в 100 долларов США. Клиент платит в евро, курс обмена на момент создания заказа 1 долл. = 0,7658 евро. Сумма платежа в евро: 76,58 евро = 100 долл.

Система Мерчанта будет уведомлена о платеже со следующими параметрами:

- `order_amount=100;`
- `order_currency=USD;`
- `balance_amount=76.58;`
- `balance_currency=EUR.`

Когда Клиент платит той же валютой, которая указана в ссылке заказа, значения `balance_amount` и `balance_currency` будут такими же, как в `order_amount` и `order_currency`:

- `order_amount=100;`
- `order_currency=USD;`
- `balance_amount=100;`
- `balance_currency=USD.`

3.7.3.3.5. Примеры запросов типа "check"

Пример запроса от системы OnPay в систему Мерчанта иллюстрирует листинг 3.54.

Листинг 3.54

```
POST https://merchant_server/script
order_amount=100.00
order_currency=USD
pay_for=123456
type=check
md5=*
```

API Мерчанта отвечает системе OnPay, формат ответа XML приведен в листинге 3.55.

Листинг 3.55

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
<code>0</code>
<pay_for>123456</pay_for>
<comment>OK</comment>
<md5>*</md5>
</result>
```

Здесь:

- code — код результата обработки операции;
- comment — обычно это описание ошибки для внутреннего использования (например, протоколирования). Значение данного поля хранится в деталях платежа, в интерфейсе Мерчанта в OnPay;
- md5 — MD5 подпись ответа системы Мерчанта для OnPay (не та же самая подпись, которая получена от OnPay!), сгенерированная из строки со следующими параметрами платежа: type;pay_for;order_amount; order_currency;code;secret_key_api_in.

Ответ системы Мерчанта об ошибке обработки содержит листинг 3.56.

Листинг 3.56

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
<code>2</code>
<pay_for>123456</pay_for>
<comment>User account doesn't exist</comment>
<md5>*</md5>
</result>
```

3.7.3.3.6. Примеры запросов типа "pay"

Пример запроса от системы OnPay в систему Мерчанта иллюстрирует листинг 3.57.

Листинг 3.57

```
POST https://merchant_server/script
onpay_id=12345
pay_for=123456
order_amount=100.00
order_currency=USD
balance_amount=76.58
balance_currency=EUR
exchange_rate=0.7658
paymentDateTime=2006-03-24T19:00:00+03:00
type=pay
md5=*
```

Ответ API системы Мерчанта системе OnPay содержит листинг 3.58.

Листинг 3.58

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
<code>0</code>
<comment>OK</comment>
<onpay_id>12345</onpay_id>
<pay_for>123456</pay_for>
<order_id>98765</order_id>
<md5>*</md5>
</result>
```

Здесь:

- `onpay_id` — ID транзакции (платежа), сохраненной в OnPay (должен быть таким же, как и в запросе от OnPay);
- `order_id` — ID транзакции (заказа), сохраненный в системе Мерчанта (опциональный, для лучшего отслеживания платежей);
- `md5` — MD5 подпись ответа системы Мерчанта для OnPay (не та же самая подпись, которая получена от OnPay!), сгенерированная из строки со следующими параметрами платежа: `type;pay_for;onpay_id;order_id;order_amount;order_currency;code;secret_key_api_in`.

3.7.3.3.7. Коды завершения операций API

Если API Мерчанта недоступно для "check"-запросов или возвращает какой-либо код кроме "0", OnPay не примет платеж от Клиента. Возможные коды завершения операции приведены в табл. 3.5.

Если API Мерчанта недоступно для "pay"-запросов, OnPay попробует достичь его повторяющимися запросами несколько раз в течение следующих 72 часов. Повторяющиеся запросы посылаются с увеличивающимися интервалами.

Пример запроса от OnPay в систему Мерчанта с ошибкой, возвращаемой из API Мерчанта, приведен в листинге 3.59.

Листинг 3.59

```
POST https://merchant_server/script
order_amount=100.00
order_currency=USD
balance_amount=76.58
balance_currency=EUR
exchange_rate=0.7658
pay_for=123456
type=check
md5=*
```

Ответ API системы Мерчанта системе OnPay приведен в листинге 3.60.

Листинг 3.60

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
<code>10</code>
<pay_for>123456</pay_for>
<comment>System in maintenance mode</comment>
<md5>*</md5>
</result>
```

Таблица 3.5. Возможные коды завершения операции

Код	Описание
0	Означает, что "уведомление о платеже принято", если тип запроса был "pay", или "может быть принято", если тип запроса был "check"
2	Только для запросов типа "check" Платеж отклонен. В этом случае OnPay не примет платеж от Клиента
3	Ошибка в параметрах. OnPay не будет пытаться повторно посыпать это уведомление в API Мерчанта и отметит этот платеж статусом "уведомление не доставлено в API", если тип запроса "pay". Если тип запроса "check", OnPay не примет этот платеж

Таблица 3.5 (окончание)

Код	Описание
7	Ошибка авторизации. MD5-подпись неверна
10	Временная ошибка. OnPay попробует повторно послать это уведомление несколько раз в течение следующих 72 часов, после чего пометит платеж статусом "уведомление не доставлено в API"

3.7.3.3.8. Обработка повторных запросов

Если API Мерчанта получает запрос с тем же `onpay_id`, как в уже успешно сохраненной транзакции, то API Мерчанта возвращает результат предыдущего сохраненного запроса (обычно это код успешного завершения, 0), т. к. неудачные транзакции не должны сохраняться в системе Мерчанта (только в протоколах).

Пример ответа на повторный запрос системы OnPay к системе Мерчанта иллюстрирует листинг 3.61.

Листинг 3.61

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
<code>0</code>
<comment>OK (существующая транзакция)</comment>
<onpay_id>123456</onpay_id>
<order_id>98765</order_id>
<md5>*</md5>
</result>
```

3.7.4. Подключение приема платежей в автоматическом режиме через ONPAY Merchant API

Скрипт подключения приема платежей в автоматическом режиме через ONPAY Merchant API находится в файле `prgoplata/onpay.php` (листинг 3.62).

Листинг 3.62

```
<?php
// подключение к файлу настроек
require_once("../my.php");
// подключение к базе данных
require_once("../mybaza.php");
// подключение к файлу формирования доступа к товарам
require_once("../arhivtovar/create_new_htaccess.php");
$key=OP_KEY;

$ff=fopen("1.txt", "w");
```

```
//Ответ на запрос check от OnPay (проверка наличия заказа в базе данных)
if($_REQUEST['type']=='check')
{
    $error=0;
    $order_amount=$_REQUEST['order_amount'];
    $order_currency=$_REQUEST['order_currency'];
    $code=$pay_for=$_REQUEST['pay_for'];
    $md5=$_REQUEST['md5'];
    $sum=floatval($order_amount);
    $code=intval($code); //Код должен быть целым числом
    //Проверка кода и переводимой за него суммы
    $txt=$order_amount.";".$order_currency.";".$code.";".$md5.";".$sum.";";
    fwrite($ff,$error.$txt);
    fclose($ff);
    //поиск заказа в базе
    $query1="SELECT * FROM zakaz WHERE id='".$code."'";
    $rez1=mysql_query($query1);
    if($rez1)
    {
        if(mysql_num_rows($rez1))
        {
            $result=answer($_REQUEST['type'],0,$pay_for,
            $order_amount,$order_currency,'OK');
            //Отвечаляем серверу OnPay, что все хорошо, можно принимать деньги
        }
        else
            $error=1;
    }
    else
        $error=1;
    if($error==1)
        $result=answer($_REQUEST['type'],2,$pay_for,
        $order_amount,$order_currency,'Error code pay_for: '.$code);
        //Сообщаем ошибку
    }
//Ответ на запрос pay от OnPay
if($_REQUEST['type']=="pay")
{
    $onpay_id=$_REQUEST['onpay_id'];
    $code=$pay_for=$_REQUEST['pay_for'];
    $order_amount=$_REQUEST['order_amount'];
    $order_currency=$_REQUEST['order_currency'];
    $balance_amount=$_REQUEST['balance_amount'];
    $balance_currency=$_REQUEST['balance_currency'];
    $exchange_rate=$_REQUEST['exchange_rate'];
```

```
$paymentDateTime=$_REQUEST['paymentDateTime'];
$md5=$_REQUEST['md5'];
$error='';
//Проверка входных данных
$txt=$onpay_id.";".$code.";".$order_amount.";".$order_currency.";
$txt.= $balance_amount.";".$balance_currency.";".$exchange_rate.";";
$txt.=$paymentDateTime.";".$md5;
if (empty($onpay_id))
{
    $error .="Не указан id<br>";
}
else
{
    if (!is_numeric(intval($onpay_id)))
        {$error .="Параметр не является числом<br>"}
}

if (empty($order_amount))
    {$error .="Не указана сумма<br>"}
else
{
    if (!is_numeric($order_amount))
        {$error .="Параметр не является числом<br>"}
}

if (empty($balance_amount))
    {$error .="Не указана сумма<br>"}
else
{
    if (!is_numeric(intval($balance_amount)))
        {$error .="Параметр не является числом<br>"}
}

if (empty($balance_currency))
    {$error .="Не указана валюта<br>"}
else
{
    if (strlen($balance_currency)>4)
        {$error .="Параметр слишком длинный<br>"}
}

if (empty($order_currency))
    {$error .="Не указана валюта<br>"}
else
{
```

```
if (strlen($order_currency)>4)
{$error .="Параметр слишком длинный<br>";}
}

if (empty($exchange_rate))
{$error .="Не указана сумма<br>";}
else
{
if (!is_numeric($exchange_rate))
{$error .="Параметр не является числом<br>";}
}
//***** //Если нет ошибок
if (!$error)
{
/**
fwrite($ff,$txt);
fclose($ff);
/**/
if(is_numeric($code))
{//Если pay_for - число
$code=intval($code); //Код должен быть целым числом
$sum=floatval($order_amount);
//Проверяем, что код есть в базе данных, и оплачиваемая сумма не меньше
допустимой
$query1="SELECT * FROM zakaz WHERE id='".$code."'";
$rez1=mysql_query($query1);
if(mysql_num_rows($rez1) == 1)
{
//Создаем строку хэша с присланных данных
$md5fb=strtoupper(md5($_REQUEST['type'].";".$pay_for.";".$onpay_id.";".$order_
amount.";".$order_currency.";".$key.""));
//Сверяем строчки хеша (присланную и созданную нами)
if ($md5fb != $md5)
{
$re-
zult=answerpay($_REQUEST['type'],7,$pay_for,$order_amount,$order_currency,'Md5
signature is wrong',$onpay_id);}
else
{
// изменение статуса заказа
$Id=$code;
require_once("mybaza.php");
$query1="UPDATE zakaz SET pay='yes'
WHERE id='".$Id."'";
$rez1=mysql_query($query1);
// создание ссылок
```

```
$data=date('Y-m-d',strtotime('now+10days'));
$query2="SELECT id,id_tovar FROM zakaz_table
        WHERE id_zakaz='".$Id."' ";
$rez2=mysql_query($query2);
while($row2=mysql_fetch_assoc($rez2))
{
$query3="SELECT arhiv FROM tovars WHERE
        id='".$row2[id_tovar]."' ";
$link=mysql_result(mysql_query($query3),0);
$query4="SELECT id_user FROM zakaz WHERE id='".$Id."'";
$id_user=mysql_result(mysql_query($query4),0);
$query5="INSERT INTO link_downloads SET id_user='".$id_user"',
        file='".$link"',status='yes',data='".$data."',
        id_zakaz='".$Id."'";
$rez5=mysql_query($query5);
$id_link=mysql_insert_id();
$query6="UPDATE zakaz_table SET id_link='".$id_link."'
        WHERE id='".$row2[id]."' ";
$rez6=mysql_query($query6);
}

// изменения в файле .htaccess
create_new_htaccess();
//*****



if ($result1)
//Если занесение информации в базу данных прошло без ошибок,
{
$result=answerpay($_REQUEST['type'],0,$pay_for,
$order_amount,$order_currency,'OK',$onpay_id);
}
else
{
$result=answerpay($_REQUEST['type'],3,$pay_for,
$order_amount,$order_currency,'Error in merchant database queries:
operation or balance tables error',$onpay_id);
}
}

else
{
$result=answerpay($_REQUEST['type'],3,$pay_for,
$order_amount,$order_currency,'Cannot find any pay
rows according to this parameters: wrong payment',$onpay_id);
}
```

```

else {
    //Если pay_for - не правильный формат
    $result=answerpay($_REQUEST['type'],3,$pay_for,
    $order_amount,$order_currency,'Error in
    parameters data',$onpay_id);
}
}

//Если есть ошибки
else
{
    fwrite($ff,$error.$txt);
    fclose($ff);
    $result=answerpay($_REQUEST['type'],3,$pay_for,
    $order_amount,$order_currency,'Error in parameters data',
    $onpay_id);
}
}

echo $result;

//Функция выдает ответ для сервиса оплаты в формате XML на руу-запрос
function answerpay($type,$code,$pay_for,$order_amount,$order_currency,$text,
$onpay_id)
{
    global $key;
    $md5=strtoupper(md5("$type;$pay_for;$onpay_id;$pay_for;$order_amount;
    $order_currency; $code;$key"));
    return "<?xml version=\"1.0\" encoding=\"UTF-8\"?\".>\n<result>\n<code>".

$code."</code>\n <comment>".$text."</comment>\n<onpay_id>".$onpay_id.
"</onpay_id>\n <pay_for>".$pay_for."</pay_for>\n<order_id>".$pay_for.
"</order_id>\n<md5>".$md5."</md5>\n</result>";
}

//Функция выдает ответ для сервиса оплаты в формате XML на чек-запрос
function answer($type,$code,$pay_for,$order_amount,$order_currency,$text)
{
    global $key;
    $md5=strtoupper(md5("$type;$pay_for;$order_amount;$order_currency;$code;$key"));
    return "<?xml version=\"1.0\" encoding=\"UTF-8\"?\".>\n<result>\n<code>".

$code."</code>\n<pay_for>".$pay_for."</pay_for>\n<comment>".$text."</comment>\n<md5>".$md5."</md5>\n</result>";
}
?>

```

При успешной оплате скрипт изменяет статус заказа в базе данных, формирует уведомления по e-mail и сервису мгновенных оповещений покупателю и администратору об успешной оплате заказа.

3.8. Блок "Заказы"

В разд. 3.6.4 мы рассматривали процесс программирования создания заказа из корзины. Теперь рассмотрим программирование следующих операций с заказами пользователя:

- просмотр заказов пользователя;
- поиск заказов;
- редактирование неоплаченных заказов;
- удаление заказов;
- получение товаров из оплаченного заказа.

3.8.1. Просмотр заказов пользователя

Пользователь может просмотреть все свои заказы при выборе в своем профиле пункта главного меню **Заказы** (рис. 3.39). При нажатии ссылки **Заказы** вызывается хайп-функция `View_All_Zakaz`, расположенная в файле `prgzakaz/view_all_zakaz.php`. Для каждого заказа выводим следующие данные:

- номер;
- дату и время заказа;
- сумму заказа;
- сумму оплаты;
- оплачено (да/нет);
- ссылки для операций с каждым заказом в зависимости от статуса заказа (посмотреть, удалить, редактировать, оплатить).

Список заказов выводится постранично. Константа `NN2` в файле настроек `my.php` определяет количество заказов на странице. Содержимое файла `prgzakaz/view_all_zakaz.php` приведено в листинге 3.63.

Листинг 3.63

```
<?php
function View_All_Zakaz($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // формирование контента
    $content=f_view_all_zakaz($Id);
    // вывод результатов
    $objResponse->assign("center3","innerHTML",$content[0]);
    // вывод навигатора страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
    // блок center3 в зону видимости
    $objResponse->script ("document.getElementById('center3').style.visibility='visible'");
}
```

```
('center3').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

Добро пожаловать, 11111
Нет сообщений

Товары **Заказы** **Переписка** **Оплата** **Выход**

Заголовок

- Все товары (10)
- Скрипты (0)
- Базы данных (7)
- Карты регионов, городов (0)
- Клипарт (3)

АКЦИЯ !!!

База предприятий Юг России 2009.
200.00 - 100.00 руб.
В корзину Подробно

База почтовых индексов населенных пунктов России
100.00 - 95.00 руб.
В корзину Подробно

Поиск заказов

Номер заказа
ИЛИ все по фильтру
с даты 2010-07-16
по дату 2010-07-17
В заказе товар (наименование)

Найти->

Все заказы

Номер	Дата	Сумма руб	Оплата руб	visible
32	2010-07-16 12:07:30	157.50	0.00	нет
31	2010-07-16 12:07:12	120.00	0.00	нет
7	2009-10-18 20:36:23	5.00	0.00	да
3	2009-10-18 09:10:24	5.00	0.00	да

Курсы валют

30.562 38.905

Корзина

Корзина пуста

У партнеров

База предприятий России
Блог по Ajax - хайп,Query
On-line телефонные справочники России
Федеральный розыск лиц на 08.2009
Розыск похищенного

Рис. 3.39. Все заказы пользователя

Контент формирует функция `f_view_all_zakaz.php`, расположенная в файле `prgzakaz/function_view_all_zakaz.php`. При этом заказы, "удаленные" пользователем, не выводятся.

ЗАМЕЧАНИЕ

Заказы, удаленные пользователем, фактически не удаляются, а сохраняются в базе данных. Просто устанавливается значение поля `visible=no` в таблице `zakaz`.

Содержимое файла `prgzakaz/function_view_all_zakaz.php` приведено в листинге 3.64.

Листинг 3.64

```
<?php
// Просмотр заказов пользователя постранично (дата по убыванию)
// $Id - номер страницы для показа
// id пользователя = $_SESSION[user]
function f_view_all_zakaz($Id)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
```

```
$text=array();
$text1="";
$query0="SELECT COUNT(id) FROM zakaz WHERE
        id_user='". $_SESSION[user]."' &&
        visible='yes' ORDER BY data DESC ";
$rez0=mysql_query($query0);
$count=mysql_result($rez0,0);
$pages=ceil($count/NN2);
$page=min($Id,$pages);$poz=($page-1)*NN2;
$text1.="

";
if($count>0)
    // список заказов непустой
{
    $query1="SELECT * FROM zakaz WHERE id_user='". $_SESSION[user]."' &&
            visible='yes' ORDER BY data DESC LIMIT ".$poz.", ".NN2."'";
    $rez1=mysql_query($query1);
    $text1.="

|                                                                                                                                                                                                                                                                                                                                                                                   |                  |                    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|--------------------|
| Номер                                                                                                                                                                                                                                                                                                                                                                             |                  |                    |
| Дата                                                                                                                                                                                                                                                                                                                                                                              |                  |                    |
| Сумма<br>руб                                                                                                                                                                                                                                                                                                                                                                      |                  |                    |
| Оплачата<br>руб                                                                                                                                                                                                                                                                                                                                                                   |                  |                    |
| </td>                                                                                                                                                                                                                                                                                                                                                                             |                  |                    |
| </td></tr>";\$i=0;     while(\$row1=mysql_fetch_assoc(\$rez1))     {         // для раскраски таблицы в "тельняшку"         \$i++;         // номер заказа         \$text1.=" <tr><td .".\$i%2+1)."'="" align="right" class="str">". \$row1[id] ."</td></tr> ";         // дата заказа         \$text1.=" <td .".\$i%2+1)."'="" align="right" class="str">". \$row1[data] ."</td> | ". \$row1[id] ." | ". \$row1[data] ." |
| ". \$row1[id] ."                                                                                                                                                                                                                                                                                                                                                                  |                  |                    |


```

```
color='red'>нет</font></td>";
$text1.=<td class='str'.($i%2+1)."' align=right">;
if($row1[pay]=='no')
// для неоплаченных - ссылки
// Редактировать и Оплатить
{
$text1.=<a href='javascript:void();' onclick=
xajax_Oplata_Zakaz("$.row1[id].");
title='Оплатить'><img src='img/pay.gif'></a>";
$text1.=<a href='javascript:void();' onclick=
xajax_Edit_Zakaz("$.row1[id].";
title='Редактировать'><img src='img/edit.png'></a>";
}
// ссылки Подробно и Удалить
$text1.=<a href='javascript:void();' onclick=
xajax_View_Zakaz("$.row1[id].";
title='Подробно'><img src='img/view.gif'></a>";
$text1.=<a href='javascript:void();' onclick=
xajax_Delete_Zakaz("$.row1[id].";
title='Удалить'><img src='img/delete.png'></a>";
$text1.= "</td></tr>";
}
$text1.= "</table>";
// список ссылок перехода по страницам
$text2="";
if($pages>1)
{
if($page != 1)
{$i=$page-1;
$text2.=<a href='javascript:void(null);' onclick=
var x=new Array();x=".($page-1).";
xajax_View_All_Zakaz(x);'> <<</a>";
}
$x=array();
$x=doarray1($page,$pages,5);
for($i=0;$i < count($x);$i++)
//for($i=1;$i <= $pages;$i++)
{
if($x[$i]==$page)
$text2.= "<a> ".$x[$i]."</a>";
else
{
$text2.=<a href='javascript:void(null);' onclick=
var x=new Array();x=". $x[$i].";
xajax_View_All_Zakaz(x);'> ".$x[$i]."</a>";
```

```
        }
    }

    if ($page != $pages)
    {$i=$page+1;
    $text2.="
```

3.8.2. Поиск заказов пользователя по фильтру

При большом количестве заказов необходим фильтр поиска заказов (см. рис. 3.39), позволяющий проводить поиск по следующим параметрам:

- по номеру заказа;
- по периоду;
- по товару в заявке (соответствию наименования).

При пустых значениях полей формы поиска, ведется поиск всех заказов пользователя. Форму поиска заказа по фильтру формирует функция `f_form_search_zakaz`, расположенная в файле `prgzakaz/function_form_search_zakaz.php` (листинг 3.65).

Листинг 3.65

```
<?php
// Форма поиска заказа
function f_form_search_zakaz()
{
require_once ("my.php");
```

```

require_once("mybaza.php");
// создаем форму
$text1.=<form id='FormSearchZakaz' action='javascript:void(null);'
onsubmit='xajax.\\"ButtonFormSearchZakaz\\").disabled=true;
xajax.\\"ButtonFormSearchZakaz\\").value=\"Подождите...\">";
xajax_View_Search_Zakaz(xajax.getFormValues(
\"FormSearchZakaz\"));'>";
$text1.=<table width=100%>;
$text1.=<tr><td width=50%>Номер заказа</td>";
// поле номер заказа
$text1.=<td width=50%>
<input type='text' name='number_zakaz' value='' size=5
maxlength=5>
// скрытое поле страницы
<input type='hidden' id='pagesearch' name='pagesearch'
value='1'></td></tr>";
$text1.=<tr><td width=50%><b>ИЛИ все по фильтру</b></td></td></tr>";
// начальная дата
$text1.=<tr><td width=50%>с даты</td>";
$text1.=<td width=50%>
<input type='text' name='datazakaz1' id='datazakaz1'
value='".date('Y-m-d')."' size=10 maxlength=10></td></tr>";
$text1.=<tr><td width=50%>по дату </td>";
// конечная дата
$text1.=<td width=50%>
<input type='text' name='datazakaz2' id='datazakaz2'
value='".date('Y-m-d', strtotime('now +1 day'))."' size=10
maxlength=10></td></tr>";
// строка наименования товара для поиска
$text1.=<tr><td width=50%>В заказе товар( наименование)</td>";
$text1.=<td width=50%>
<input type='text' name='name_tovar' value='' size=20
maxlength=20></td></tr>";
$text1.=</table>";
$text1.=<center><br><input type='submit' id='ButtonFormSearchZakaz'
value='Найти ->'></center>";
$text1.=</form>;
return $text1;
}
?>

```

При нажатии на кнопку **Найти** вызывается хайп-функция `View_Search_Zakaz`, в качестве входных параметров передаются значения формы. Содержимое файла `prgzakaz/view_search_zakaz.php`, в котором находится функция `View_Search_Zakaz`, приведено в листинге 3.66.

Листинг 3.66

```
<?php
function View_Search_Zakaz ($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // заголовок блока
    $zag=f_zag1("Результат поиска заказов");
    $objResponse->assign("centercaption3","innerHTML",$zag);
    // формирование результатов поиска
    $content=f_view_search_zakaz ($Id);
    $objResponse->assign("center3","innerHTML",$content[0]);
    // навигатор страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
    // блок center3 в зону видимости
    $objResponse->script ("document.getElementById
        ('center3').scrollIntoView();");
    // активировать кнопку выбора
    $objResponse->assign("ButtonFormSearchZakaz","value", "Найти ->");
    $objResponse->assign("ButtonFormSearchZakaz","disabled", false);
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Поиск результатов поиска товаров по фильтру иллюстрирует рис. 3.40. Поиск результатов поиска и формирование контента происходит в функции `f_view_search_zakaz.php`, расположенной в файле `prgzakaz/function_view_search_zakaz.php`. Для поиска заказов, содержащих выбранный товар (по совпадению наименования), будем использовать сложные запросы.

ЗАМЕЧАНИЕ

Опыт показывает, что сложные запросы выполняются медленно, и лучше применить несколько простых запросов, чем один сложный. Но в нашем случае это нарушит стройность процедуры, поэтому будем использовать сложный запрос.

Сложный запрос нам необходим для поиска товара по наименованию в таблицах `zakaz_table`, `tovars` и таблице `zakaz`.

Пример запроса приведен в листинге 3.67.

Листинг 3.67

```
$query0="SELECT DISTINCT(zakaz.id) FROM zakaz,zakaz_table,tovars WHERE
    zakaz.visible='yes' && zakaz.id_user='".$_SESSION[user]."' ";
$name_tovar=utftowin($Id[name_tovar]);
```

```

if($Id[number_zakaz]>0)
{
$query0.=&& zakaz.id='".$Id[number_zakaz]."' ";
}
else
{
$query0.=&& zakaz.data >= '".$Id[datazakaz1]."' && zakaz.data
<='".$Id[datazakaz2]."' ";
if(strlen(rtrim(ltrim($name_tovar)))>0)
{
$query0.=&& LOWER(tovars.name) LIKE '%'.$name_tovar.'%' &&
zakaz_table.id_tovar=tovars.id &&
zakaz_table.id_zakaz=zakaz.id ";
}
}
}

```

Номер	Дата	Сумма руб	Оплата руб	
31	2010-07-16 12:07:12	120.00	0.00	нет

Рис. 3.40. Результат поиска заказов по фильтру

Особенность вывода строк для блока "Заказы" — необходимость формировать ссылки в зависимости от статуса заказа на просмотр, редактирование, удаление, переход на оплату заказа. Содержимое файла prgzakaz/function_view_search_zakaz.php приведено в листинге 3.68.

Листинг 3.68

```

<?php
function f_view_search_zakaz($Id)
{
require_once("my.php");
require_once("mybaza.php");

```

```
$text=array();
$text1="";

$query0="SELECT DISTINCT(zakaz.id) FROM zakaz,zakaz_table,tovars
        WHERE zakaz.visible='yes' &&
              zakaz.id_user='".$SESSION[user]."' ";
$name_tovar=utftown($Id[name_tovar]);
if($Id[number_zakaz]>0)
{
    $query0.= "&& zakaz.id='".$Id[number_zakaz]."' ";
}
else
{
    $query0.= "&& zakaz.data >= '".$Id[datazakaz1]."' &&
              zakaz.data <='".$Id[datazakaz2]."' ";
    if(strlen(rtrim(trim($name_tovar)))>0)
    {
        $query0.= "&& LOWER(tovars.name) LIKE '%".$name_tovar."%' &&
                  zakaz_table.id_tovar=tovars.id
                  && zakaz_table.id_zakaz=zakaz.id ";
    }
}

$rez0=mysql_query($query0);
$count=mysql_num_rows($rez0);
$pages=ceil($count/NN2);
$page=min($Id[pagesearch],$pages);$poz=($page-1)*NN2;
$text1=<div class='zag_view_tovars'>";
if($count>0)
{
    $query0.=" ORDER BY zakaz.data DESC LIMIT ".$poz.", ".NN2."";
    $query1=$query0;
    $rez1=mysql_query($query1);
    $text1.= "<table>";
    $text1.= "<tr><td class='str0' align=right>Home</td>";
    $text1.= "<td class='str0'>Дата</td>";
    $text1.= "<td class='str0' align=right>Сумма<br>pyб</td>";
    $text1.= "<td class='str0' align=right>Оплата<br>pyб</td>";
    $text1.= "<td class='str0'></td>";
    $text1.= "<td class='str0'></td></tr>";$i=0;
    while($row1=mysql_fetch_row($rez1))
    {
        $i++;
        $query2="SELECT * FROM zakaz WHERE id='".$row1[0]."' ";
        $rez2=mysql_query($query2);
```

```

$row2=mysql_fetch_assoc($rez2);
$text1.= "<tr><td class='str'." . ($i%2+1) . "'"
        align=right>".$row2[id]."</td>";
$text1.= "<td class='str'." . ($i%2+1) . "'>".$row2[data]."</td>";
$text1.= "<td class='str'." . ($i%2+1) . "'"
        align=right>".$row2[summa_rub]."</td>";
$text1.= "<td class='str'." . ($i%2+1) . "'"
        align=right>".$row2[summa_rub_oplata]."</td>";
if($row2[pay]=='yes')
    $text1.= "<td class='str'." . ($i%2+1) . "' align=right><font"
    color='blue'>да</font></td>";
else
    $text1.= "<td class='str'." . ($i%2+1) . "' align=right><font"
    color='red'>нет</font></td>";
$text1.= "<td class='str'." . ($i%2+1) . "' align=right>";
if($row2[pay]=='no')
{
    $text1.= "<a href='javascript:void()' onclick='"
        xajax_Oplata_Zakaz(\".$row2[id]\");'
        title='Оплатить'><img src='img/pay.gif'></a>";
    $text1.= "<a href='javascript:void()' onclick='"
        xajax_Edit_Zakaz(\".$row2[id]\");'
        title='Редактировать'><img src='img/edit.png'></a>";
}
$text1.= "<a href='javascript:void()' onclick='"
        xajax_View_Zakaz(\".$row2[id]\");'
        title='Помощь'><img src='img/view.gif'></a>";
$text1.= "<a href='javascript:void()' onclick='"
        xajax_Delete_Zakaz(\".$row2[id]\");'
        title='Удалить'><img src='img/delete.png'></a>";
$text1.= "</td></tr>";
}
$text1.= "</table>";
// список ссылок перехода по страницам
$text2="";
if($pages>1)
{
if($page != 1)
{$i=$page-1;
$text2.= "<a href='javascript:void(null)' onclick='"
    document.forms.FormSearchZakaz.pagesearch.value=\"$i.\";"
    xajax_View_Search_Zakaz(xajax.getFormValues(
    \"FormSearchZakaz\"));'> <<</a>";
}
}

```

```
$x=array();
$x=doarray1($page,$pages,5);
for($i=0;$i < count($x);$i++)
//for($i=1;$i <= $pages;$i++)
{
if($x[$i]==$page)
$text2.= "<a> ".$x[$i]."</a>";
else
{
$text2.= "<a href='javascript:void(null);' onclick='
document.forms.FormSearchZakaz.pagesearch.value='".$x[$i].";
xajax_View_Search_Zakaz(xajax.getFormValues(
\"FormSearchZakaz\"));'> ".$x[$i]."</a>";
}
}
if($page != $pages)
{$i=$page+1;
$text2.= "<a href='javascript:void(null);' onclick='
document.forms.FormSearchZakaz.pagesearch.value='".$i."';
xajax_View_Search_Zakaz(xajax.getFormValues(
\"FormSearchZakaz\"));'> >></a>";
}
if($pages != 1)
{$text2.= "<br><br>Всего - ".$count." Страниц -
".$pages."<br> </center>";}
else
{$text2.= "</center>";}
}
else
{
$text2=<br><center>По данному запросу поиска ничего не
обнаружено</center><br>";
}
$text[0]=$text1;
$text[1]=$text2;
return $text;
}
?>
```

3.8.3. Редактирование заказа

Пользователь может редактировать неоплаченные заказы. При нажатии иконки **Редактировать** в строке заказа, вызывается хайп-функция `Edit_Zakaz`, расположенная в файле `prgzakaz/edit_zakaz.php` (листинг 3.69). В качестве аргумента пере-

дается ID заказа в таблице `zakaz`. Из таблицы `table_zakaz`читываются позиции заказа. Форму редактирования заказа (рис. 3.41) формирует функция `f_edit_zakaz`, расположенная в файле `prgzakaz/function_edit_zakaz.php` (листинг 3.70).

The screenshot shows a web-based order viewing interface. At the top, a blue header bar reads 'Просмотр заказа'. Below it, a title 'Заказ 31' is displayed. A status message 'Статус - неоплачен' follows. The main content is a table showing the details of the order:

	Товар	Кол-во	Цена, руб	Сумма, руб
1	База почтовых индексов населенных пунктов России	1	47.50	47.50
2	Единый федеральный реестр туроператоров	1	22.50	22.50
3	База предприятий Юг России 2009.	1	50.00	50.00
Итого		3		120.00

Below the table are two buttons: 'Изменить ->' (Edit) and 'Назад' (Back). The entire form is contained within a light gray box.

Рис. 3.41. Форма редактирования заказа

Листинг 3.69

```
function Edit_Zakaz($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // формирование формы редактирования
    $content=f_edit_zakaz($Id);
    // заголовок
    $zagcontent=f_zag1("Просмотр заказа");
    // вывод заголовка
    $objResponse->assign("centercaption5","innerHTML",$zagcontent);
    // вывод формы
    $objResponse->assign("center5","innerHTML",$content);
    // блок center5 в зону видимости
    $objResponse->script("document.getElementById
        ('center5').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
```

Листинг 3.70

```
<?php
function f_edit_zakaz($Id)
{
    // подключение файла настроек
    require_once("mybaza.php");
    // подключение к базе данных
    require_once("my.php");
    $text1="";
    $text1.=<form id='FormEditZakaz' action='javascript:void(null);'
        onsubmit='
            xajax.$(\"ButtonFormEditZakaz\").disabled=true;
            xajax.$(\"ButtonFormEditZakaz\").value=\"Подождите...\";
            xajax_Go>Edit_Zakaz(xajax.getFormValues(
                \"FormEditZakaz\"));\'>;
    $text1.=<table width=100%>;
    $query1="SELECT * FROM zakaz WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    // номер заказа
    $text1.=<br><br><b>Заказ</b> ".$row1[id]."<br>";
    // статус
    if($row1[pay]=='yes')
        $text1.=<br><b>Статус </b> - оплачен<br>;
    else
        $text1.=<br><b>Статус </b> - неоплачен<br>;
    $query2="SELECT * FROM zakaz_table WHERE id_zakaz='".$Id."'";
    $rez2=mysql_query($query2);
    // шапка для таблицы позиций товара
    $text1.=<table>;
    $text1.=<tr><td class='str0'></td>;
    $text1.=<td class='str0'>Товар</td>;
    $text1.=<td class='str0' align=right>Кол-во</td>;
    $text1.=<td class='str0' align=right>Цена,<br>руб</td>;
    $text1.=<td class='str0' align=right>Сумма,<br>руб</td>;
    $text1.=<td class='str0'></td>;
    $text1.=</td></tr>;
    $i=0;$count=0;
    while($row2=mysql_fetch_assoc($rez2))
    {
        // номер позиции для зебры
        $i++;
        // накопление суммы
        $count+=$row2[kol];
```

```

$text1.=<tr>;
$text1.=<td class='str' .($i%2+1)."' width=5%>".$i."
    <input type=hidden name=table_id".$i." id=table_id".$i."
        value='".$row2[id]."' >
    </td>";
$query1="SELECT name FROM tovars WHERE id='".$row2[id_tovar]."' ";
$text1.=<td class='str' .($i%2+1)."' width=60%>
    ".mysql_result(mysql_query($query1),0)."/>";
// количество и формирование вызова Change_Kol_Table по onchange
// READONLYZ не равно readonly
$text1.=<td class='str' .($i%2+1)."' width=10%><input type=text
    name=table_kol".$row2[id]."
    id=table_kol".$row2[id].". value='".$row2[kol]."'
    size=3 maxlength=3 ".READONLYZ."
    onchange='var x=new Array();
    x[0]='".$row2[id].";x[1]=this.value;
    x[2]=document.forms.FormEditZakaz.table_pay
    ".$row2[id].".value;
    x[3]=document.forms.FormEditZakaz.table_summa
    ".$row2[id].".value;
    x[4]=document.forms.FormEditZakaz.itogo_count_zakaz.value;
    x[5]=document.forms.FormEditZakaz.itogo_summa_zakaz.value;
    xajax_Change_Kol_Table(x);'
    >/>";
// цена
$text1.=<td class='str' .($i%2+1)."' width=10%><input type=text
    name=table_pay".$row2[id]."
    id=table_pay".$row2[id].". value='".$row2[pay_rub]."'
    size=3 maxlength=6 readonly
    onclick='document.getElementById(
    \'table_kol".$row2[id]."\').focus();
    return false;'>/>";
// сумма для позиции
$text1.=<td class='str' .($i%2+1)."' width=10%><input type=text
    name=table_summa".$row2[id]."
    id=table_summa".$row2[id].". value='".$row2[summa_rub]."'
    size=6 maxlength=6 readonly
    onclick='document.getElementById(
    \'table_kol".$row2[id]."\').focus();
    return false;'>/>";
// удалить позицию, если запрещено редактирование
if(READONLYZ=='readonly')
    $text1.=<td class='str' .($i%2+1)."' width=5%><a href='
        javascript:void();' onclick='
        var x=new Array();x[0]='".$row2[id].";x[1]=\"0\";

```

```
x[2]=document.forms.FormEditZakaz.table_pay  
".$row2[id].".value;  
x[3]=document.forms.FormEditZakaz.table_summa  
".$row2[id].".value;  
x[4]=document.forms.FormEditZakaz.itogo_count_zakaz.value;  
x[5]=document.forms.FormEditZakaz.itogo_summa_zakaz.value;  
xajax_Change_Kol_Table(x);'><img src='img/delete.png'>  
</a></td>;  
$text1.= "</tr>";  
}  
$text1.= "<tr><td></td>";  
// вывод итого  
$text1.= "<td>Итого</td>";  
$text1.= "<td align=right><input type=text name=itogo_count_zakaz  
id=itogo_count_zakaz  
value='".$count."' size=8 maxlength=8 readonly></td>";  
$text1.= "<td align=right></td>";  
$text1.= "<td align=right><input type=text name=itogo_summa_zakaz  
id=itogo_summa_zakaz  
value='".$row1[summa_rub]."' size=8 maxlength=8  
readonly></td>";  
$text1.= "<td></td></tr>";  
$text1.= "</table>";  
// кнопки Изменить и Возврат  
$text1.= "<center><input type='submit' id='ButtonFormEditZakaz'  
value='Изменить ->'>  
<br><br><input type='button' value='Назад' onclick='  
document.getElementById(\"center5\").innerHTML=\"\";  
document.getElementById(\"centercaption5\").innerHTML=  
\"\";';></center>";  
$text1.= "</form>";  
return $text1;  
}  
?  

```

Для каждого заказа можно отредактировать количество заказываемого товара. При изменении поля по событию onchange вызывается хайп-функция Change_Kol_Table, которая пересчитывает новую сумму для позиции и итоговую сумму для всего заказа. Функция Change_Kol_Table находится в файле prgzakaz/change_kol_table.php (листинг 3.71).

Листинг 3.71

```
<?php  
// Изменение кол-ва товара  
// в позиции товара при редактировании заказа
```

```

function Change_Kol_Table($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // вычислить И установить новое значение в ячейку суммы
    $new_summa=$Id[1]*$Id[2];
    $objResponse->assign("table_summa".$Id[0],"value",$new_summa);
    // вычислить изменение количества и суммы
    $kol_old=$Id[3]/$Id[2];
    $delta_kol=$Id[1]-$kol_old;
    $delta_summa=$delta_kol*$Id[2];
    $new_kol=$kol_old+$delta_kol;
    $objResponse->assign("table_kol".$Id[0],"value",$new_kol);
    // вычислить новые ИТОГО количества и суммы И установить
    $new_itogo_kol=$Id[4]+$delta_kol;
    $new_itogo_summa=$Id[5]+$delta_kol*$Id[2];
    $objResponse->assign("itogo_count_zakaz","value",$new_itogo_kol);
    $objResponse->assign("itogo_summa_zakaz","value",$new_itogo_summa);
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?
```

ЗАМЕЧАНИЕ

Константа `READONLYZ` в файле настроек `my.php` отвечает за возможность редактирования количества в форме редактирования заказа пользователя (рис. 3.42). При значении `READONLYZ='readonly'` изменение количества товара при редактировании заказа запрещено, в этом случае присутствует иконка удаления для удаления позиции товара (рис. 3.43).

Просмотр заказа					
Заказ 31					
Статус - неоплачен					
	Товар	Кол-во	Цена, руб	Сумма, руб	
1	База почтовых индексов населенных пунктов России	<input type="text" value="1"/>	<input type="text" value="47.50"/>	<input type="text" value="47.50"/>	
2	Единый федеральный реестр туроператоров	<input type="text" value="2"/>	<input type="text" value="22.50"/>	<input type="text" value="45"/>	
3	База предприятий Юг России 2009.	<input type="text" value="1"/>	<input type="text" value="50.00"/>	<input type="text" value="50.00"/>	
Итого		<input type="text" value="4"/>		<input type="text" value="142.5"/>	
Изменить ->					
Назад					

Рис. 3.42. Изменение количества позиции товара при редактировании заказа

Просмотр заказа

Заказ 31

Статус - неоплачен

Товар	Кол-во	Цена, руб	Сумма, руб
1 База почтовых индексов населенных пунктов России	1	47.50	47.50
2 Единый федеральный реестр туроператоров	1	22.50	22.50
3 База предприятий Юг России 2009.	1	50.00	50.00
Итого	3		120.00

Изменить ->

Назад

Рис. 3.43. Форма редактирования товара при `readonlyz='readonly'`

Для сохранения изменений отредактированного заказа данные формы `id='FormEditZakaz'` отправляются функции `Go>Edit_Zakaz`, расположенной в файле `prgzakaz/go_edit_zakaz.php`, которая записывает измененные данные для отредактированного заказа в базу. Данные по каждой позиции заносятся в таблицу `zakaz_table`, новая сумма — в таблицу `zakaz`. При удачном изменении отображается сообщение **Изменено** и выводится измененное содержимое заказа (рис. 3.44). Содержимое файла `prgzakaz/go_edit_zakaz.php` приведено в листинге 3.72.

Листинг 3.72

```
<?php
// Сохранение отредактированного заказа
function Go>Edit_Zakaz ($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // вычислить количество строк
    $j=(count($Id)-2)/4;
    for($i=1;$i<=$j;$i++)
    {
        // id позиции
        $id=$Id[table_id.$i];
        // количество
        $kol=$Id[table_kol.$id];
        // сумма позиции
        $summa_rub=$Id[table_summa.$id];
```

```

// изменение данных позиции в таблице zakaz_table
$query1="UPDATE zakaz_table SET
        kol='".$kol."',summa_rub='".$summa_rub."'
        WHERE id='".$id."' ";
$rez1=mysql_query($query1);
}
// успешное изменение
$objResponse->alert("Изменено !!!");
// изменение общей суммы заказа в таблице zakaz
$query2="SELECT id_zakaz FROM zakaz_table WHERE id='".$id."'";
$id_zakaz=mysql_result(mysql_query($query2),0);
$query3="UPDATE zakaz SET summa_rub='".$Id[itogo_summa_zakaz]."'"
        WHERE id='".$id_zakaz."'";
mysql_query($query3);
// формирование нового контента
$content=f_edit_zakaz($id_zakaz);
// формирование заголовка
$zagcontent=f_zag1("Просмотр заказа");
// вывод заголовка
$objResponse->assign("centercaption5","innerHTML",$zagcontent);
// вывод контента
$objResponse->assign("center5","innerHTML",$content);
// блок center5 в зону видимости
$objResponse->script("document.getElementById
('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

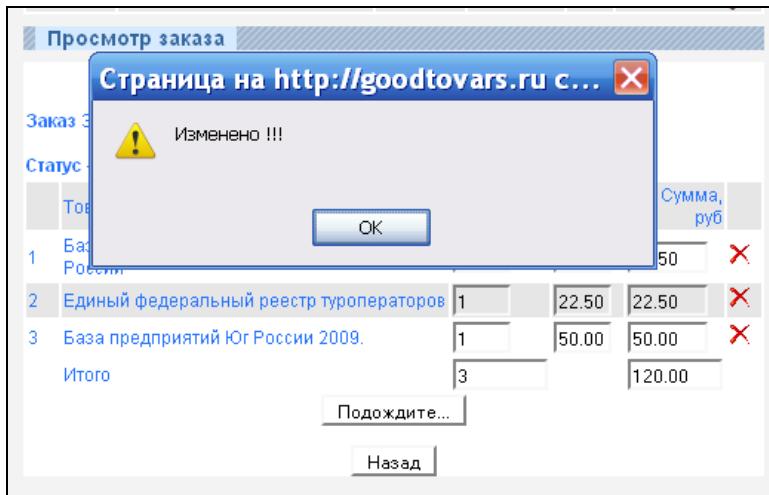


Рис. 3.44. Редактирование заказа прошло успешно

3.8.4. Просмотр заказа

Пользователь может просмотреть свой заказ. При нажатии иконки **Просмотр** в строке заказа, вызывается хайп-функция `View_Zakaz`, расположенная в файле `prgzakaz/view_zakaz.php` (листинг 3.73). В качестве аргумента передается ID заказа в таблице `zakaz`. Из таблицы `table_zakaz`читываются позиции заказа. Для оплаченных заказов напротив каждого товара выводим ссылку на страницу скачивания (рис. 3.45).

The screenshot shows two pages from a web application. The top page is titled 'Все заказы' (All orders) and lists four orders in a table:

Номер	Дата	Сумма руб	Оплата руб	Статус	Actions
32	2010-07-16 12:07:30	157.50	0.00	нет	
31	2010-07-16 12:07:12	120.00	0.00	нет	
7	2009-10-18 20:38:23	5.00	0.00	да	
3	2009-10-18 09:10:24	5.00	0.00	да	

The bottom page is titled 'Просмотр заказа' (Order details) and shows the details for Order #7:

Заказ 7
Статус - оплачен

Товар	Кол-во	Цена, руб	Сумма, руб
1 Справочник калорийности продуктов	1	5.00	5.00
Итого	1		5.00

Рис. 3.45. Просмотр заказа

Листинг 3.73

```
<?php
// Просмотр заказа
// $Id - id заказа
function View_Zakaz($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // формирование контента
    $content=f_view_zakaz($Id);
    // формирование заголовка
    $zagcontent=f_zag1("Просмотр заказа");
    // вывод заголовка
    $objResponse->assign("centercaption5","innerHTML",$zagcontent);
    // вывод контента
    $objResponse->assign("center5","innerHTML",$content);
```

```
// блок center5 в зону видимости
$objResponse->script ("document.getElementById
    ('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

Контент просмотра заказа формирует функция `f_view_zakaz`, расположенная в файле `prgzakaz/function_view_zakaz.php` (листинг 3.74).

Листинг 3.74

```
<?php
// Просмотр заказа
// $Id - id заказа
function f_view_zakaz($Id)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    $text=array();
    $text1="";
    // поиск заказа в базе данных
    $query1="SELECT * FROM zakaz WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    // номер заказа
    $text1.= "<br><br><b>Заказ</b> ".$row1[id]."<br>";
    // статус
    if($row1[pay]=='yes')
        $text1.= "<b>Статус </b> - оплачен<br>";
    else
        $text1.= "<b>Статус </b> - неоплачен<br><br>";
    // поиск позиций в базе
    $query2="SELECT * FROM zakaz_table WHERE id_zakaz='".$Id."'";
    $rez2=mysql_query($query2);
    // вывод шапки для позиций
    $text1.= "<table>";
    $text1.= "<tr><td class='str0'></td>";
    $text1.= "<td class='str0'>Товар</td>";
    $text1.= "<td class='str0' align=right>Кол-во</td>";
    $text1.= "<td class='str0' align=right>Цена,<br>руб</td>";
    $text1.= "<td class='str0' align=right>Сумма,<br>руб</td>";
```

```
$text1.=<td class='str0'></td></tr>";
$i=0;
$count=0;
while($row2=mysql_fetch_assoc($rez2))
{
    $i++;
    $text1.= "<tr><td class='str'." . ($i%2+1) . "' align=right>".$i."</td>";
    $query3="SELECT name FROM tovars WHERE id='".$row2[id_tovar]."' ";
    $rez3=mysql_query($query3);
    // наименование
    $text1.= "<td class='str'." . ($i%2+1) . "'>
    ".mysql_result($rez3,0)."</td>";
    // количество
    $text1.= "<td class='str'." . ($i%2+1) . "' align=right>
    ".$row2[kol]."</td>";
    $count+=$row2[kol];
    // цена
    $text1.= "<td class='str'." . ($i%2+1) . "' align=right>
    ".$row2[pay_rub]."</td>";
    // сумма
    $text1.= "<td class='str'." . ($i%2+1) . "' align=right>
    ".$row2[summa_rub]."</td>";
    // ссылка на скачивание, если есть (для оплаченных)
    if($row2[id_link]>0)
        $text1.= "<td class='str'." . ($i%2+1) . "'><a href='download.php
        ?id=".$row2[id_link]."' target='_blank'>
        <img src='img/download.png'></a></td></tr>";
    else
        $text1.= "<td class='str'." . ($i%2+1) . "'></td>";
    $text1.= "</tr>";
}
$text1.= "<tr><td></td></tr>";
$text1.= "<td>Итого</td>";
// итого количество
$text1.= "<td align=right>".$count."</td>";
$text1.= "<td align=right></td>";
// итого сумма
$text1.= "<td align=right>".$row1[summa_rub]."</td>";
$text1.= "<td></td></tr>";
if($row1[pay]=='no')
{
    $text1.= "<tr><td></td>";
    $text1.= "<td>Оплачено</td>";
    $text1.= "<td align=right></td>";
    $text1.= "<td align=right></td>";
    $text1.= "<td align=right>".$row1[summa_rub_oplata]."</td>";
```

```

$text1.= "<td></td></tr>";
}
$text1.= "</table>";
// для неоплаченного заказа - ссылка на Оплатить
if($row1[pay]=='no')
    $text1.= "<br><br><input type=button value='Оплатить' onclick='
        xajax_Oplata_Zakaz(\".$row1[id].\");' >";
return $text1;
}

?>

```

3.8.5. Удаление заказа

Пользователь может удалить свой заказ. При нажатии иконки **Удалить** в строке заказа, вызывается хайп-функция `Delete_Zakaz`, расположенная в файле `prgzakaz/delete_zakaz.php` (листинг 3.75). В качестве аргумента передается ID заказа в таблице `zakaz`. При успешной операции удаления выводится сообщение **Заказ удален** и выводится обновленный список заказов пользователя (рис. 3.46).

Листинг 3.75

```

<?php
// Удаление заказа
// visible=no
function Delete_Zakaz($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value", 'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // "удаление" заказа
    $query1="UPDATE zakaz SET visible='no' WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    if($rez1)
    {
        // сообщение об успешном удалении
        $objResponse->alert("Заказ удален !!!");
        // сформировать обновленный список всех заказов на страницу 1
        $content=f_view_all_zakaz(1);
        // вывести контент
        $objResponse->assign("center3","innerHTML",$content[0]);
        // навигатор страниц
        $objResponse->assign("center4","innerHTML",$content[1]);
        // блок center5 в зону видимости
        $objResponse->script("document.getElementById

```

```

('center3').scrollIntoView();";
// очистить блоки center5 и centercaption5
$objResponse->assign("centercaption5",
    "innerHTML", "<table></table>");
$objResponse->assign("center5", "innerHTML", "<table></table>");
}
else
// ошибка удаления
$objResponse->alert("Ошибка удаления !!!");
$objResponse->assign("flag_ajax", "value", 'no');
return $objResponse;
}
?>

```

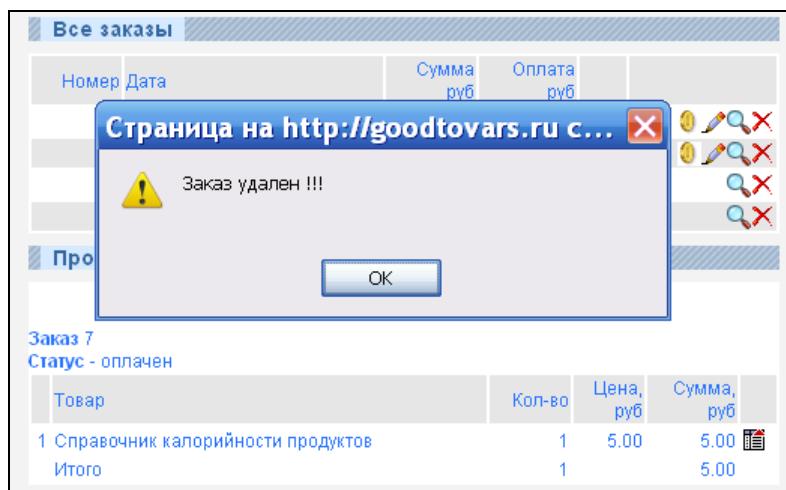


Рис. 3.46. Удаление заказа

ЗАМЕЧАНИЕ

Удалять заказ полностью из базы данных мы не будем. В таблице `zakaz` мы предусмотрели поле `visible`. Для скрытия заказа от просмотра достаточно установить значение этого поля, равное `no`.

3.8.6. Оплата заказа. Формирование ссылок для скачивания

Все цифровые товары выдаются в виде файлов архива `zip`, `tar` и `gz`. Подробнее с этим познакомимся, когда будем рассматривать форму добавления товара на панели администратора. Все архивы при создании товара складываются в папку `arhivtovar`. Естественно, необходимо ограничить доступ к файлам этой папки. Будем использовать файлы `.htaccess` (механизм опишем в разд. 3.8.7). Сейчас рассмотрим

формирование списка ссылок на файлы при оплате заказа пользователем либо оплате заказа администратором.

При успешной оплате заказа через сервис Webmoney (файл prgoplatा/wm_result.php) или ONPAY (файл prgoplatа/onpay.php) в таблице link_downloads базы данных для каждой строки делаем следующие действия: из таблицы tovars получаем путь к файлу товара, в таблицу link_downloads делаем запись с данными ссылки и ID новой записи в link_downloads записываем для каждого товара в zakaz_table (поле id_link). Программная реализация приведена в листинге 3.76.

Листинг 3.76

```
<?php
$Id=$id_zakaz;
// получение позиций заказа
$query2="SELECT id,id_tovar FROM zakaz_table
        WHERE id_zakaz='".$Id."'";
$rez2=mysql_query($query2);
while($row2=mysql_fetch_assoc($rez2))
{
    // получение пути к файлу товара
    $query3="SELECT arhiv FROM tovars WHERE id='".$row2[id_tovar]."' ";
    $link=mysql_result(mysql_query($query3),0);
    // получение пользователя, сделавшего заказ
    $query4="SELECT id_user FROM zakaz WHERE id='".$Id."'";
    $id_user=mysql_result(mysql_query($query4),0);
    // записать в таблицу link downloads данные
    // для записи
    // id_user
    // файл товара
    // id_zakaz
    // дату
    $query5="INSERT INTO link_downloads SET id_user='".$id_user"',
            file='".$link."',status='yes',
            data='".$data."',id_zakaz='".$Id."'";
    $rez5=mysql_query($query5);
    // получить id записи
    $id_link=mysql_insert_id();
    // и записать в строку позиции товара для
    // данного заказа
    $query6="UPDATE zakaz_table SET id_link='".$id_link."'
            WHERE id='".$row2[id]."' ";
    $rez6=mysql_query($query6);
}
?>
```

3.8.7. Регулирование доступа к файлам скачивания с использованием файла .htaccess

Файл .htaccess (обратите внимание, что первый символ в названии файла — точка) применяется для управления Web-сервером Apache со стороны конечного пользователя хостинга. Вы помещаете в этот файл директивы, которые Web-сервер воспринимает и обрабатывает, выполняя далее действия в соответствии с настройками, которые были сделаны пользователем.

Файл .htaccess можно разместить в корневом каталоге Web-сервера (прямо в каталоге public_html) и в этом случае директивы из такого файла действуют по всему Web-серверу. Файл .htaccess может находиться и в конкретном подкаталоге сервера, и тогда директивы, которые указаны в этом файле, "перекрывают" действие директив из "основного" файла, который размещен в каталоге public_html или в любом каталоге более высокого уровня. Иными словами, действие директив из .htaccess наследуется сверху вниз, но не наоборот. Изменения, внесенные в файл, вступают в силу немедленно. Это связано с тем, что информация из .htaccess перечитывается при каждом обращении к Web-серверу Apache.

В .htaccess можно поместить большинство из доступных директив для Web-сервера. Следует заметить, что директивы, в описании которых в поле Context отсутствует упоминание .htaccess, недоступны для использования в этом файле конфигурации.

Как запретить Web-посетителям читать файлы в каталоге?

Запрет на все файлы:

```
deny from all
```

Здесь all обозначает "все".

Разрешить доступ только с определенного IP:

```
order allow deny  
deny from all  
allow from ip_адрес
```

В данном случае, ip_адрес обозначает конкретный адрес, например:

```
order allow deny  
deny from all  
allow from 192.126.12.199
```

Запретить доступ для определенного IP:

```
order allow deny  
deny from all  
deny from ip_адрес
```

Назначение параметра ip_адрес аналогично ранее приведенному примеру.

Запрет доступа на группу файлов по маске иллюстрирует листинг 3.77.

Листинг 3.77

```
<Files "\.(inc|sql|...другие расширения...) $">  
order allow,deny
```

```
deny from all
</Files>
```

Определяет доступ к файлу по его расширению. Например запрет на доступ к файлам с расширениям "inc" для Web-посетителей:

```
<Files "\.(inc)$">
order allow,deny
deny from all
</Files>
```

В данном примере сам Web-сервер Apache может обращаться к файлам с таким расширением, в отличие от посетителей сайта.

Можно поставить запрет на конкретный файл по его названию и расширению (листинг 3.78).

Листинг 3.78

```
Files config.inc.php>
order allow,deny
deny from all
</Files>
```

В данном примере установлен запрет на обращения к файлу config.inc.php для посетителей сайта, сам же сервер может использовать этот файл.

Нам потребуются записи для каждого файла (листинг 3.79).

Листинг 3.79

```
<Files "1254513156.rar">
order deny,allow
deny from all
allow from 77.39.66.172,127.0.0.1,194.186.219.71,194.186.219.132
</Files>
<Files "1254511325.rar">
order deny,allow
deny from all
allow from 77.39.66.172,194.186.219.71
</Files>
```

3.8.8. Получение товара

При нажатии на ссылку получения товара мы переходим на страницу <http://MYSITE/download.php?id=33>, открытую в другом окне. Здесь значение ID — id_link в таблице link_download. Файл download.php (листинг 3.80) выдает ссылку для скачивания (рис. 3.47). При ограничении по времени действия ссылки выдается сообщение (рис. 3.48). Если произвольный пользователь попробует набрать эту ссылку в браузере, то эта попытка получить файл закончится неудачей (рис. 3.49).

Листинг 3.80

```
<html>
<head>
    <title>Скачивание оплаченного товара</title>
</head>
<body width=200px height=200px>
<?php
// подключение файла настроек
require_once("my.php");
// подключение к базе данных
require_once("mybaza.php");
// подключение программы пересоздания файла .htaccess

require_once("arhivtovar/create_new_htaccess.php");
// формирование нового .htaccess
create_new_htaccess();
// получение данных ссылки
$query1="SELECT * FROM link_downloads WHERE id_link='".$_GET[id]."' ";
$rez1=mysql_query($query1);
$file=mysql_result($rez1,0,"file");
$id_user=mysql_result($rez1,0,"id_user");
$status=mysql_result($rez1,0,"status");
$data=mysql_result($rez1,0,"data");
if($status=='no')
    // невозможно - ограничение по дате
    $text1="Скачивание невозможно - допуск до ".$data;
else
{
    // формирование ссылки
    $query2="SELECT ip FROM users WHERE id='".$id_user."'";
    $rez2=mysql_query($query2);
    $ip=mysql_result($rez2,0);
    $ip=str_replace(";", "<br>", $ip);
    $text1=<center>;
    $text1.="Ссылка для скачивания - <a href='".$SITE.$file."'>";
    $text1.=SITE.$file."</a><br>";
    $text1.="Ссылка действительна до ".$data."<br>";
    $text1.="Ссылка действительна для ip-адресов :<br>".$ip;
    $text1.=</center>;
}
echo $text1;
?>
</body>
</html>
```

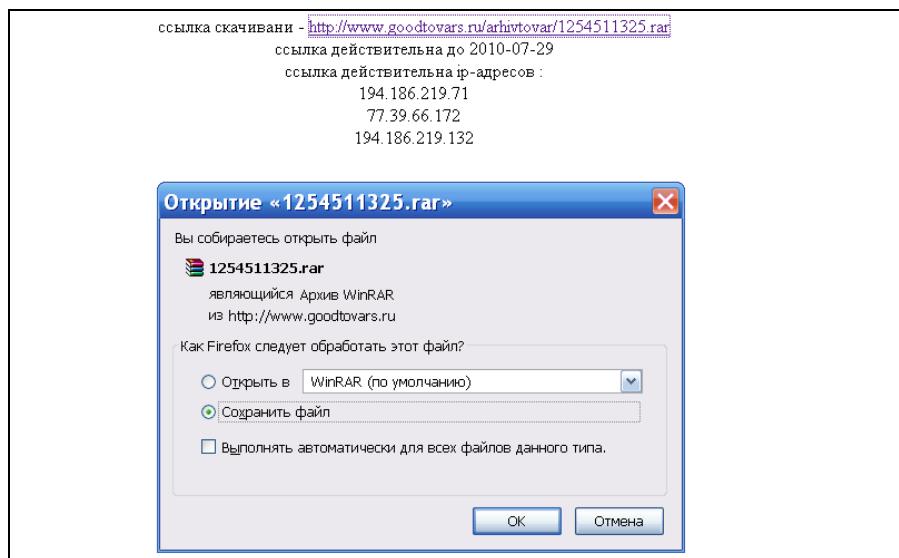


Рис. 3.47. Получение ссылки на скачивание товара

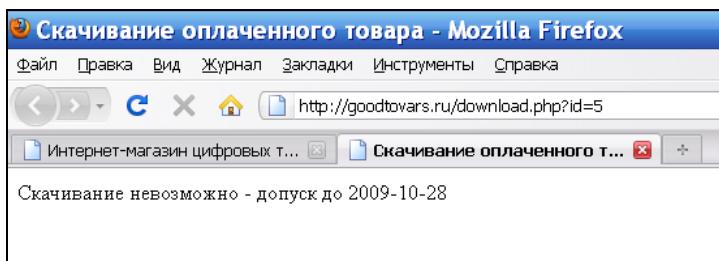


Рис. 3.48. Превышение временного интервала на скачивание

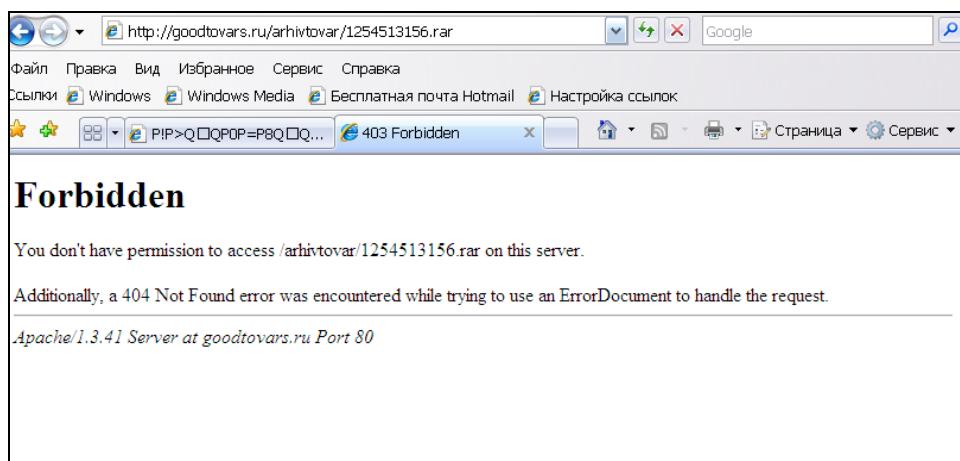


Рис. 3.49. Закрыт доступ к файлу по ссылке для других пользователей

Из файла download.php каждый раз вызывается функция `create_new_htaccess`, расположенная в файле arhivtovar/create_new_htaccess.php (листинг 3.81), которая пересоздает файл .htaccess. Такая необходимость связана с тем, что зарегистрированный пользователь может войти с другого IP-адреса (при входе в профиль в список адресов пользователя (таблица users поле ip) заносится новый IP).

Листинг 3.81

```
<?php
function create_new_htaccess()
{
    // подключение к базе данных
    require_once("mybaza.php");
    // файлы из директории arhivtovar
    $dir=opendir("arhivtovar");
    while($file1=readdir($dir))
    {
        if($file1 != "." && $file1 != ".." && $file1 != ".htaccess" &&
           $file1 != "create_new_htaccess.php" && $file1 != "index.php")
        {
            // есть пользователи, имеющие доступ к файлу скачивания ?
            // (в таблице link_downloads)
            $query1="SELECT DISTINCT(id_user) FROM link_downloads WHERE
                    file='arhivtovar/".$file1."' && status='yes' ";
            $rez1=mysql_query($query1);
            if(mysql_num_rows($rez1)>0)
            {
                // да, есть - открыть доступ к файлу для их ip
                $textfiles.= "<Files ".$file1.">
                                order deny,allow
                                deny from all
                                allow from ";
                while($row1=mysql_fetch_row($rez1))
                {
                    $query2="SELECT ip FROM users WHERE id='".$row1[0]."' ";
                    $rez2=mysql_query($query2);
                    $ip=mysql_result($rez2,0);
                    $ip=str_replace(";",",",$ip);
                    $textfiles.=$ip;
                }
            }
            closedir($dir);
            $ff=fopen("arhivtovar/.htaccess","w");
            $textfiles.=$ip;
            fwrite($ff,$textfiles);
            fclose($ff);
        }
    }
}
```

```
// записать новое содержимое .htaccess  
fwrite($ff,$textfiles);  
fclose($ff);  
//  
return str_replace("\r\n","<br>",$textfiles);  
}  
  
?>
```

3.9. Блок мгновенных сообщений на сайте

Сейчас на многих сайтах появилась функция мгновенных сообщений пользователю, даже если он не обновляет страницу, например сообщение от другого пользователя. Реализуем мы такой функционал на нашем сайте.

3.9.1. Вывод мгновенных сообщений

Блок для вывода мгновенных сообщений находится в правом верхнем углу страницы сайта, чуть ниже блока входа. JavaScript-функция `time_go` с периодичностью, установленной в переменной `time_pr=10000` (в данном случае 10 с) обращается к серверу с запросом новых сообщений для текущего пользователя. Содержимое функции `time_go` приведено в листинге 3.82.

Листинг 3.82

```
<script type="text/javascript">  
function time_go()  
{  
var time_pr=10000;  
setTimeout("time_go()",time_pr);  
if(document.getElementById("flag_ajax").value=="no")  
{  
var str1=<img src='img/load.gif'> загрузка;  
document.getElementById("header4").innerHTML=str1;  
xajax_Timer1(document.getElementById("flag_time1").value);  
}  
return;  
}  
</script>
```

Из листинга 3.82 видно, что если не установлен флаг вызова хайп-функции, скрипт вызывает хайп-функцию `xajax_Timer1`, расположенную в файле `prgmessageheader4/timer1.php` (листинг 3.83). В качестве аргумента передается ID текущего пользователя, хранящийся в поле ввода `flag_time1`. Функция выводит количество непросмотренных сообщений для данного пользователя и ссылку на первое непро-

смотренное сообщение вида: Само сообщение (data) (рис. 3.50).

ЗАМЕЧАНИЕ

Непрочитанными сообщениями в блоке мгновенных сообщений для пользователя являются сообщения, для которых в таблице message_header4 базы данных значение поля visible=yes. При переходе по ссылке мгновенного сообщения, значение поля visible устанавливается равным no для этого сообщения.

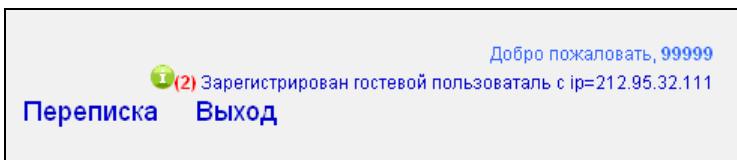


Рис. 3.50. Блок мгновенных сообщений

Листинг 3.83

```
<?php
// Новые сообщения на
// главной странице
function Timer1($Id)
{
    session_start();
    $_SESSION[user]=$_SESSION[user];
    $_SESSION[session]=$_SESSION[session];
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // выбор сообщений для пользователя
    $query1="SELECT COUNT(id) FROM message_header4
            WHERE id_user='".$Id."' && visible='yes' ";
    $count=mysql_result(mysql_query($query1),0);
    if($count>0)
    {
        // есть сообщение
        // выбираем первое непрочитанное
        $query2="SELECT message FROM message_header4
                WHERE id_user='".$Id."' && visible='yes' ORDER BY id ASC LIMIT 1";
        // формируем ссылку
        $new_messages=mysql_result(mysql_query($query2),0);
        $new_messages=<img src='img/notification.gif'>".<font
```

```

color=red><b>(".$count.")</b></font> ".$new_messages;
}
else
    // нет сообщений
    $new_messages='Нет сообщений';
// вывести в блок header4
$objResponse->assign("header4","innerHTML",$new_messages);
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

3.9.2. Переход по ссылке мгновенных сообщений

При нажатии на сообщение-ссылку вызывается хайп-функция `ajax_Go_Timer1`, расположенная в файле `prgmessageheader4/go_timer1.php` (листинг 3.84). Задача функции — по значению `type` и `arg_id` вызвать соответствующую функцию (просмотр пользователя, просмотр внутреннего сообщения, просмотр заказа), передав ей аргумент `arg_id`. Кроме того, необходимо пометить как прочитанное это сообщение (установить поле `visible=yes`) и вывести следующее сообщение либо надпись **Нет сообщений** при его отсутствии.

Листинг 3.84

```

<?php
function Go_Timer1($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // получить данные ссылки
    $query3="SELECT id_user,type,arg_id  FROM message_header4
        WHERE id='".$Id."'";
    $rez3=mysql_query($query3);
    $id_user=mysql_result($rez3,0,"id_user");
    $type=mysql_result($rez3,0,"type");
    $arg_id=mysql_result($rez3,0,"arg_id");
    switch($type)
    {
        // переход на форму регистрации
        case 1: $content=f_form_reg_user();
            $zagcontent=f_zag1("Регистрация");
            $objResponse->assign("centercaption2",
                "innerHTML",$zagcontent);
            $objResponse->assign("center2","innerHTML",$content);

```

```
$objResponse->script("document.getElementById  
    ('center2').scrollIntoView();");  
break;  
// список пользователей для админа  
case 2: $content=f_view_user_admin($arg_id);  
    $zagcontent=f_zag1("Просмотр данных пользователя");  
    $objResponse->assign("centercaption5",  
        "innerHTML",$zagcontent);  
    $objResponse->assign("center5","innerHTML",$content);  
    $objResponse->script("document.getElementById  
        ('center5').scrollIntoView();");  
    break;  
case 3:  
    break;  
// посмотреть сообщение внутренней почты  
case 4: $content=f_view_message($arg_id);  
    $zagcontent=f_zag1("Просмотр сообщения");  
    $objResponse->assign("centercaption5",  
        "innerHTML",$zagcontent);  
    $objResponse->assign("center5","innerHTML",$content);  
    $objResponse->script("document.getElementById  
        ('center5').scrollIntoView();");  
    break;  
// просмотр заказа админ  
case 5: $content=f_view_zakaz_admin($arg_id);  
    $zagcontent=f_zag1("Просмотр заказа");  
    $objResponse->assign("centercaption5",  
        "innerHTML",$zagcontent);  
    $objResponse->assign("center5","innerHTML",$content);  
    $objResponse->script("document.getElementById  
        ('center5').scrollIntoView();");  
    break;  
// просмотр оплаченного заказа админ  
case 6: $content=f_view_zakaz_admin($arg_id);  
    $zagcontent=f_zag1("Просмотр заказа");  
    $objResponse->assign("centercaption5",  
        "innerHTML",$zagcontent);  
    $objResponse->assign("center5","innerHTML",$content);  
    $objResponse->script("document.getElementById  
        ('center5').scrollIntoView();");  
    break;  
default:  
    break;  
}  
// сделать сообщение прочтеным
```



```
// 6-ссылка заказ (поступившую оплату)
// $arg2 - id заказа, сообщения
// $arg3 - id_user (кому)
// $arg4 - текст
function f_create_message_header4($arg1,$arg2,$arg3,$arg4)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    $type=$arg1;
    $arg_id=$arg2;
    $id_user=$arg3;
    $data=date('Y-m-d H:i:s');
    // добавить сообщение в базу данных
    $query0="INSERT INTO message_header4 SET data='".$data."',
        id_user='".$id_user."'";
    mysql_query($query0);
    // получить id нового сообщения
    $id=mysql_insert_id();
    // формируем ссылку
    switch($arg1)
    {
        //
        case 1: $message='<a href="javascript:void();" onclick="'
            xajax_Go_Timer1('.$id.');"'.$arg4.'</a>';
            break;
        case 2: $query01="SELECT id FROM users WHERE type='9' ";
            $id_user=mysql_result(mysql_query($query01),0);
            $message='<a href="javascript:void();" onclick="'
            xajax_Go_Timer1('.$id.');"'.$arg4.'</a>';
            break;
        case 3:
            break;
        case 4: $message='<a href="javascript:void();" onclick="'
            xajax_Go_Timer1('.$id.');"'.$arg4.'</a>';
            break;
        case 5: $query01="SELECT id FROM users WHERE type='9' ";
            $id_user=mysql_result(mysql_query($query01),0);
            $message='<a href="javascript:void();" onclick="'
            xajax_Go_Timer1('.$id.');"'.$arg4.'</a>';
            break;
        case 6: $message='<a href="javascript:void();" onclick="'
            xajax_Go_Timer1('.$id.');"'.$arg4.'</a>';
            break;
```

```

    }
    // добавляем данные в запись
    $query1="UPDATE message_header4 SET id_user='".$id_user."',
        type='".$type."',arg_id='".$arg_id."',message='".$message."',
        visible='yes'      WHERE id='".$id."'  ";
    mysql_query($query1);
    return ;
}
?>

```

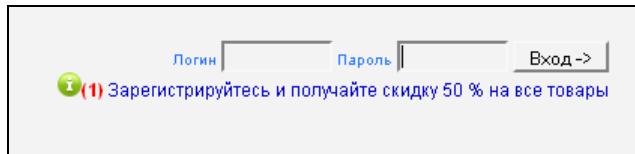


Рис. 3.51. Сообщение при первом заходе на сайт (для пользователя)

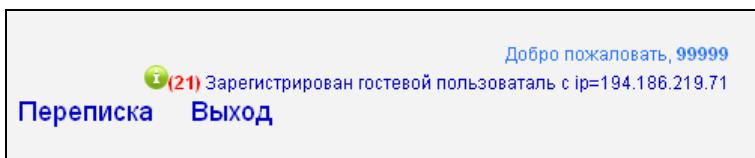


Рис. 3.52. Сообщение о регистрации пользователя (для админа)



Рис. 3.53. Сообщение о получении сообщения по внутренней почте (для получателя)

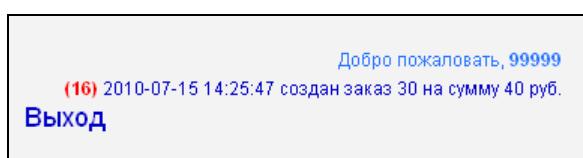


Рис. 3.54. Сообщение о созданном заказе (для админа и пользователя)

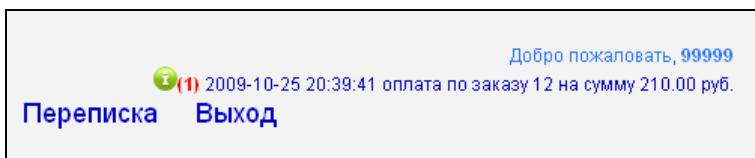


Рис. 3.55. Сообщение об оплаченном заказе (для админа и пользователя)

3.10. Переписка на сайте (внутренняя почта)

Для отправки сообщений администратору сайта созданы программы переписки на сайте. Пользователь может:

- отправлять сообщения администратору;
- получать сообщения;
- смотреть сообщения списком и по отдельности;
- удалять сообщения.

3.10.1. Просмотр сообщений пользователя списком

При выборе пункта меню **Сообщения**, открывается постраничный список сообщений (рис. 3.56) по фильтру Входящие/Исходящие. При нажатии ссылки **Входящие** вызывается хайп-функция `View_All_Messages_In`, расположенная в файле `prgmessage/view_all_messages_in.php` (листинг 3.86). При нажатии ссылки **Исходящие** вызывается хайп-функция `View_All_Messages_Out`, расположенная в файле `prgmessage/view_all_messages_out.php` (листинг 3.87). Для каждого сообщения выводим следующие данные:

- дата сообщения;
- отправитель;
- тема;
- ссылки **Просмотр** и **Удаление**.

Листинг 3.86

```
<?php
// Просмотр всех входящих сообщений постранично
// $Id - номер страницы
function View_All_Messages_In($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // функция формирования контента
    $content=f_view_all_messages_in($Id);
    // вывод контента
    $objResponse->assign("center3","innerHTML",$content[0]);
    // навигатор страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
    // блок center3 в зону видимости
    $objResponse->script("document.getElementById
        ('center3').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse();
}
?>
```

Листинг 3.87

```

<?php
// Просмотр всех исходящих сообщений постранично
// $Id - номер страницы
function View_All_Messages_Out ($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // функция формирования контента
    $content=f_view_all_messages_out($Id);
    // вывод контента
    $objResponse->assign("center3","innerHTML",$content[0]);
    // навигатор страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
    // блок center3 в зону видимости
    $objResponse->script("document.getElementById
        ('center3').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>

```

Входящие сообщения	
Входящие	Исходящие
Дата 2009-10-13 12:20:22	От 99999 Тема fdfdf
2009-10-12 12:32:02	99999 fdf
2009-10-12 12:21:54	99999 65656
2009-10-12 11:38:24	99999 erererer
2009-10-12 11:01:46	99999 9999
2009-10-08 16:14:56	99999 test2
2009-10-08 16:14:04	99999 test1

Написать администратору

Рис. 3.56. Список сообщений пользователя (входящие)

Функции формирования контента для входящих и исходящих сообщений `f_view_all_messages_in` и `f_view_all_messages_out` расположены в файлах `prgmessage/function_view_all_messages_in.php` (листинг 3.88) и `prgmessage/function_view_all_messages_out.php` (листинг 3.89) соответственно. Вывод происходит по-

странично, количество сообщений на странице — константа `NN3` в файле настроек `my.php`.

Листинг 3.88

```
<?php
// Просмотр входящих сообщений пользователя постранично
// $Id - номер страницы для показа
// id пользователя = $_SESSION[user]
function f_view_all_messages_in($Id)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    $text=array();
    $text1="";
    // выбор входящих сообщений
    $query0="SELECT COUNT(id) FROM messages WHERE
        id_to='".$SESSION[user]."' &&
        visible_to='yes' ORDER BY data DESC ";
    $rez0=mysql_query($query0);
    $count=mysql_result($rez0,0);
    $pages=ceil($count/NN3);
    $page=min($Id,$pages); $poz=($page-1)*NN3;
    // формируем ссылку на исходящие
    $text1.= "<table><tr><td align=center><b>Входящие</b>
        </td><td align=center>
        <a href='javascript:void();' onclick='
            xajax_View_All_Messages_Out(1);'
            Исходящие</a></td></tr></table>";
    $text1.= "<div class='zag_view_messages'>";
    if($count>0) // есть сообщения
    {
        $query1="SELECT * FROM messages WHERE id_to='".$SESSION[user]."'"
            && visible_to='yes' ORDER BY data DESC
            LIMIT ".$poz.", ".NN3."'";
        $rez1=mysql_query($query1);
        // шапка таблицы списка
        $text1.= "<table>";
        $text1.= "<tr><td class='str0'></td>";
        $text1.= "<td class='str0'>Дата</td>";
        $text1.= "<td class='str0'>От</td>";
        $text1.= "<td class='str0'>Тема</td>";
        $text1.= "<td class='str0'></td></tr>";
```

```
$i=0; // для раскраски в зебру
while($row1=mysql_fetch_assoc($rez1))
{
$ii++;
if($row1[view_to]=='no')
// иконка для непрочтенных
$text1.="|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| '><img src='img/new_message.gif'></td>"; else $text1.="|  |  |  |  |  | | --- | --- | --- | --- | --- | | ' width=5%></td>"; // дата $text1.=" '>".$row1[data]."</td>"; $query2="SELECT login FROM users WHERE id='".$row1[id_from]."' "; $rez2=mysql_query($query2); $login=mysql_result($rez2,0); $login=(strlen($login)<13)?($login):(substr($login,0,12)."...")); $text1.=" '>".$login."</td>"; // тема $text1.=" '>".$row1[theme]."</td>"; // иконки-ссылки Смотреть и Удалить $text1.=" '><a href='javascript:void();' onclick='xajax_View_Message(\".$row1[id].\");' title='Смотреть'><img src='img/view.gif'></a>"; $text1.=" | | | | | |

```

```

else
{
    $text2.="

```

Листинг 3.89

```

<?php
// Просмотр исх сообщений пользователя постранично (дата по убыванию)
// $Id - номер страницы для показа
// id пользователя = $_SESSION[user]

```

```
function f_view_all_messages_out($Id)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    $text=array();
    $text1="";
    // выбор исходящих сообщений
    $query0="SELECT COUNT(id) FROM messages WHERE
        id_from='".$SESSION[user]."' &&
        visible_from='yes' ORDER BY data DESC ";
    $rez0=mysql_query($query0);
    $count=mysql_result($rez0,0);
    $pages=ceil($count/NN3);
    $page=min($Id,$pages); $poz=($page-1)*NN3;
    // формируем ссылку на входящие
    $text1.= "<table><tr><td align=center><a href='javascript:void()'
        onclick='xajax_View_All_Messages_In(1);'>Входящие</a></td>
        <td align=center><b>Исходящие</b></td></tr></table>";
    $text1.= "<div class='zag_view_messages'>";
    if($count>0)
    {
        // есть сообщения
        $query1="SELECT * FROM messages WHERE id_from='".$SESSION[user]."'"
            && visible_from='yes' ORDER BY data DESC
            LIMIT ".$poz.", ".NN3."";
        $rez1=mysql_query($query1);
        // шапка списка исходящие
        $text1.= "<table>";
        $text1.= "<tr><td class='str0'></td>";
        $text1.= "<td class='str0'>Дата</td>";
        $text1.= "<td class='str0'>Кому</td>";
        $text1.= "<td class='str0'>Тема</td>";
        $text1.= "<td class='str0'></td></tr>";
        $i=0; //для раскраски в зебру
        while($row1=mysql_fetch_assoc($rez1))
        {
            $i++;
            if($row1[view_out]=='no')
                // иконка для непрочтенных
                $text1.= "<tr><td class='str'.".$i%2+1.". '><img
                    src='img/noview_message.gif'></td>";
            else
                $text1.= "<tr><td class='str'.".$i%2+1.". ' width=5%></td>";
            // дата
        }
    }
}
```

```
$text1.= "<td class='str' .($i%2+1)." '>".$row1[data]."</td>";  
$query2="SELECT login FROM users WHERE id='".$row1[id_to]." ' ";  
$rez2=mysql_query($query2);  
$login=mysql_result($rez2,0);  
$login=(strlen($login)<13)?($login):(substr($login,0,12)."....");  
$text1.= "<td class='str' .($i%2+1)." '>".$login."</td>";  
// тема  
$text1.= "<td class='str' .($i%2+1)." '>".$row1[theme]."</td>";  
// иконки-ссылки Смотреть и Удалить  
$text1.= "<td class='str' .($i%2+1)." '><a href='javascript:void()' ;'  
        onclick='xajax_View_Message(\".$row1[id].\" );'  
        title='Смотреть'><img src='img/view.gif'></a>";  
$text1.= "<a href='javascript:void()' ;'  
        onclick='xajax_Delete_Message_Out(\".$row1[id].\" );'  
        title='Удалить'><img src='img/delete.png'></a>";  
$text1.= "</td></tr>";  
}  
$text1.= "</table>";  
// список ссылок перехода по страницам  
$text2="";  
if($pages>1)  
{  
    if($page != 1)  
    {$i=$page-1;  
    $text2.= "<a href='javascript:void(null)' onclick='  
        var x=new Array();x=".($page-1).";  
        xajax_View_All_Messages_Out(x);'> <<</a>";  
    }  
    $x=array();  
    $x=doarray1($page,$pages,5);  
    for($i=0;$i < count($x);$i++)  
    {  
        if($x[$i]==$page)  
        $text2.= "<a> ".$x[$i]."</a>";  
        else  
        {  
        $text2.= "<a href='javascript:void(null)' onclick='  
            var x=new Array();x=". $x[$i].";  
            xajax_View_All_Messages_Out(x);'> ".$x[$i]."</a>";  
        }  
    }  
    if($page != $pages)  
    {$i=$page+1;  
    $text2.= "<a href='javascript:void(null)' onclick='  
        var x=new Array();x=".($page+1).";  
        xajax_View_All_Messages_Out(x);'> ".$x[$i]."</a>";  
    }  
}
```

```

        xajax_View_All_Messages_Out($x);'> >></a>";
    }
    if ($pages != 1)
    {$text2.= "<br><br>Всего - ".$count." Страниц - ".$pages."<br>
        </center>";}
    else
    {$text2.= "</center>";}
}
}
else
{
$text1.= "<center><br>Входящих сообщений нет<br></center>";
}
$text1.= "<center><br><br><a href='javascript:void()' onclick='
        xajax_Form_New_Message();'> Написать
        администратору</a></center>";
$text[0]=$text1;
$text[1]=$text2;
return $text;
}
?>

```

3.10.2. Просмотр сообщения

Просмотр сообщения происходит при нажатии на иконку **Смотреть** для данного сообщения (рис. 3.57). При этом вызывается хаах-функция `View_Message` с аргументом, равным ID сообщения в базе данных. Функция `View_Message` расположена в файле `prgmessage/view_message.php` (листинг 3.90).

Листинг 3.90

```

<?php
// Просмотр сообщения
// $Id - id message
function View_Message($Id)
{
$objResponse = new xajaxResponse();
$objResponse->assign("flag_ajax","value",'yes');
// формирование контента
$content=f_view_message($Id);
// заголовок
$zagcontent=f_zag1("Просмотр сообщения");
// вывод заголовка
$objResponse->assign("centercaption5","innerHTML",$zagcontent);
// вывод контента
$objResponse->assign("center5","innerHTML",$content);

```

```
// блок center5 в зону видимости
$objResponse->script("document.getElementById
    ('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

The screenshot shows a user interface for managing messages. At the top, there are navigation links: Товары, Заказы, Переписка, Оплата, and Выход. A message header indicates it's a personal message from user 99999 (2010-07-20 10:39:58). Below the header, there are sections for 'Входящие сообщения' (Incoming messages) and 'Исходящие' (Outgoing). The 'Входящие' section lists several messages with their dates, senders, and subjects. To the right, there are sidebar modules for 'Курсы валют' (Currency rates), 'Корзина' (Cart) which is empty, and 'У партнеров' (Partners) which includes links to various resources like the 'База предприятий России'. At the bottom, there are sections for 'Просмотр сообщения' (Message preview) and 'Статистика' (Statistics).

Дата	От	Тема
2010-07-20 10:39:58	99999	Тестовое сообщение
2009-10-13 12:20:22	99999	fdfdf
2009-10-12 12:32:02	99999	fdf
2009-10-12 12:21:54	99999	65656
2009-10-12 11:38:24	99999	erererer
2009-10-12 11:01:46	99999	9999
2009-10-08 16:14:56	99999	test2
2009-10-08 16:14:04	99999	test1

Рис. 3.57. Просмотр сообщения пользователя

Функция формирования контента `f_view_message` расположена в файле `prgmessage/function_view_message.php` (листинг 3.91).

Листинг 3.91

```
<?php
// Просмотр сообщения
// $Id - id message
function f_view_message($Id)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    $text=array();
    $text1="";
```

```

// выбор сообщения
$query1="SELECT * FROM messages WHERE id='".$Id."'";
$rez1=mysql_query($query1);
$row1=mysql_fetch_assoc($rez1);
if($_SESSION[user]==$row1[id_to]) // входящее
{
    // установка признака просмотра
    $query3="UPDATE messages SET view_to='yes' WHERE id='".$Id."'";
    mysql_query($query3);
    // установка признака просмотра для header4
    $query2="SELECT login FROM users WHERE id='".$row1[id_from].'";
    $rez2=mysql_query($query2);
    $text1.= "<br><br><b>Сообщение от ".mysql_result($rez2,0)."</b><br>";
}
else // исходящее
{
    $query2="SELECT login FROM users WHERE id='".$row1[id_to].'";
    $rez2=mysql_query($query2);
    $text1.= "<br><br><b>Сообщение для ".mysql_result($rez2,0)."</b><br>";
}
$text1.= "<br><b>Тема : </b> ".$row1[theme]."<br>";
$text1.= "<br><b>Сообщение : </b> ".$row1[message]."<br>";
// ссылка на создание сообщения
if($_SESSION[user]==$row1[id_to])
{
    if($_SESSION[type]=='1' || $_SESSION[type]=='2')
        $text1.= "<center><br><br><a href='javascript:void();' onclick='
            xajax_Form_New_Message();'> Написать администратору
            </a></center>";
    else
        $text1.= "<center><br><br><a href='javascript:void();' onclick='
            xajax_Form_New_Message_Admin(\".$row1[id_from].\");'
            Ответить </a></center>";
}
return $text1;
}
?>

```

3.10.3. Удаление сообщения

Сообщение удаляется при нажатии на иконку **Удалить** для данного сообщения (рис. 3.58). При этом вызывается хайп-функция `Delete_Message_In` (для входящего) или `Delete_Message_Out` (для исходящего) с аргументом, равным ID сообщения в базе данных. Функция `Delete_Message_In` расположена в файле `prgmessage/delete_message_in.php` (листинг 3.92), функция `Delete_Message_Out` — в файле `prgmessage/delete_message_out.php` (листинг 3.93). Удаление сообщения из базы

не происходит, в базе данных в таблице `messages` устанавливаем значение поля `visible_to` (для получателя) или `visible_from` (для отправителя) равным `no`, в зависимости от того, кто (получатель или отправитель) удаляет сообщение.

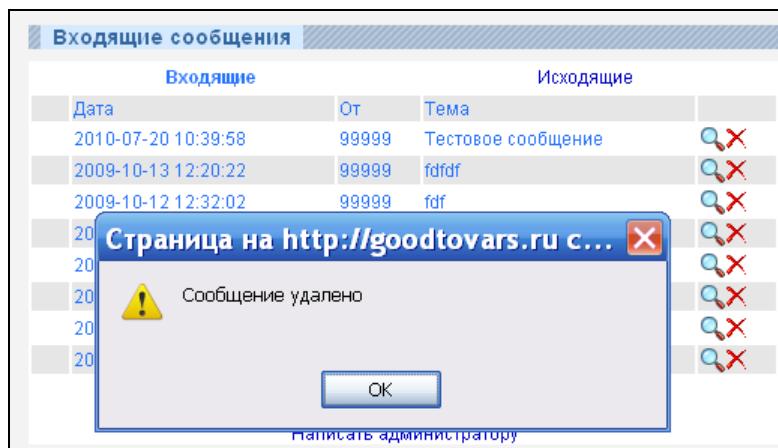


Рис. 3.58. Удаление сообщения пользователя

Листинг 3.92

```
<?php
// Удаление входящего сообщения
// $Id - id message
// установить visible_to=no
function Delete_Message_In($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // изменение записи (установить visible_to=no)
    $query1="UPDATE messages SET visible_to='no' WHERE id='".$Id."'";
    if(!mysql_query($query1))
        $objResponse->alert("Ошибка удаления сообщения !!!");
    else
        $objResponse->alert("Сообщение удалено");
    // получить обновленный список
    $content=f_view_all_messages_in($Id);
    // вывести контент
    $objResponse->assign("center3","innerHTML",$content[0]);
    // навигатор страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
}
```

```
// блок center3 в зону видимости
$objResponse->script("document.getElementById
    ('center3').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

Листинг 3.93

```
<?php
// Удаление исходящего сообщения
// $Id - id message
// установить visible_out=no
function Delete_Message_Out($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // изменение записи (установить visible_out=no)
    $query1="UPDATE messages SET visible_out='no' WHERE id='".$Id."'";
    if(!mysql_query($query1))
        $objResponse->alert("Ошибка удаления сообщения !!!");
    else
        $objResponse->alert("Сообщение удалено");
    // получить обновленный список
    $content=f_view_all_messages_out($Id);
    // вывести контент
    $objResponse->assign("center3","innerHTML",$content[0]);
    // навигатор страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
    // блок center3 в зону видимости
    $objResponse->script("document.getElementById
        ('center3').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

3.10.4. Создание сообщения

При нажатии на ссылку **Написать администратору** выводится форма составления сообщения администратору (вызов хайп-функции `Form_New_Message`) (рис. 3.59), которая расположена в файле `prgmessage/form_new_message.php` (листинг 3.94).

Листинг 3.94

```
<?php
// Форма создания нового сообщения
function Form_New_Message()
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    $text1="";
    $text1.=<form id='FormNewMessage' action='javascript:void(null)';
        onsubmit=
            xajax.\("ButtonFormNewMessage\").disabled=true;
            xajax.\("ButtonFormNewMessage\").value=
            \"Подождите...\";
            xajax_Add_New_Message(xajax.getFormValues(
                \"FormNewMessage\"));
        >';
    $text1.=<table width=100%>;
    $text1.=<caption>Форма создания сообщения</caption>;
    $text1.=<tr><td width=30%>Тема</td>
        <td width=70%><input type='text' name='theme'
            size='30' maxlength='30' value='' ></td></tr>;
    $text1.=<tr><td width=30%>Сообщение (до 256 символов)</td>
        <td width=70%><textarea name='message'
            cols='30' rows='5' value=''
        </textarea></td></tr>;
    $text1.=<tr><td></td>
        <td>
            <input type='submit' id='ButtonFormNewMessage'
            value='Создать ->' >
        </td>
        <td></td></tr>;
    $text1.=</table>;
    $text1.=</form>;
// заголовок
$zagcontent=f_zag1("Создание нового сообщения");
// вывод заголовка
$objResponse->assign("centercaption5","innerHTML",$zagcontent);
// вывод контента
$objResponse->assign("center5","innerHTML",$text1);
// блок center3 в зону видимости
$objResponse->script("document.getElementById
    ('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

Создание нового сообщения

Форма создания сообщения

Тема

Сообщение (до 256 символов)

Создать ->

Рис. 3.59. Форма создания нового сообщения

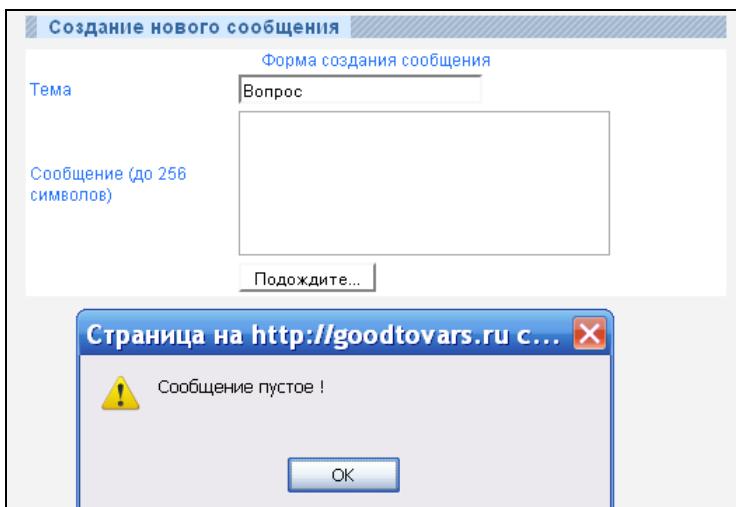


Рис. 3.60. Ошибка при создании нового сообщения

Дата	От	Тема		
2010-07-20 10:39:58	99999	Тестовое сообщение		
2009-10-13 12:20:22	99999	fdfdf		
2009-10-12 12:32:02	99999	fdf		
2009-10-12 12:21:54	99999	65656		
2009-10-12 11:38:24	99999	erererer		
2009-10-12 11:01:46	99999	9999		
20				
20				

Страница на <http://goodtovars.ru> с...

Сообщение добавлено!

OK

Рис. 3.61. Новое сообщение создано

После заполнения полей и нажатия кнопки **Создать**, происходит вызов функции `Add_New_Message` с передачей ей значений формы с `id=FormNewMessage`. Функция проверяет заполнения полей и либо выводит ошибку (рис. 3.60), либо отправляет сообщение (рис. 3.61). Функция `Add_New_Message` расположена в файле `prgmessage/add_new_message.php` (листинг 3.95).

Листинг 3.95

```
<?php
// Добавление нового сообщения
// $Id - id message
function Add_New_Message ($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе
    require_once ("mybaza.php");
    $error="";
    // проверка полей на заполнение
    if(strlen(trim($Id[theme]))==0)
        $error.="Не заполнена тема !";
    if(strlen(trim($Id[message]))==0)
        $error.="Сообщение пустое !";
    if($error=="") // отправить сообщение
    {
        $data=date('Y-m-d H:i:s');
        $id_from=$_SESSION[user];
        $query1="SELECT id FROM users WHERE type='9' ";
        $rez1=mysql_query($query1);
        $id_to=mysql_result($rez1,0);
        $theme=utftowin($Id[theme]);
        $message=utftowin($Id[message]);
        $message=str_replace("*","", $message);
        $message=str_replace("*","", $message);
        $query2="INSERT INTO messages SET data='".$data."',
            id_from='".$id_from."',id_to='".$id_to."',
            theme='".$theme."',message='".$message."'";
        if(!mysql_query($query2))
            $objResponse->alert("Ошибка добавления сообщения !");
        else
        {
            // для блока мгновенных сообщений
            $query3="SELECT login FROM users WHERE id='".$id_from."'";
            $rez3=mysql_query($query3);
            f_create_message_header4(4,mysql_insert_id(),$id_to, "Личное
```

```
сообщение от ".mysql_result($rez3,0)." (".$data.")";  
// очистить форму  
$objResponse->assign("centercaption5","innerHTML","");
$objResponse->assign("center5","innerHTML","");
$objResponse->script("document.getElementById  
('center3').scrollIntoView();");
$objResponse->alert("Сообщение добавлено!");
}  
}  
}  
else // ошибка
{
$objResponse->alert($error);
$objResponse->assign("ButtonFormNewMessage","value", "Вход ->");
// активировать кнопку
$objResponse->assign("ButtonFormNewMessage","disabled", false);
}
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}  
?  
>
```



Глава 4

Программирование панели администратора

Для управления сайтом необходимо создать панель администратора и запрограммировать для нее следующий функционал:

- управление товарами:
 - создание новых товаров;
 - удаление товаров;
 - редактирование товаров;
 - скрытие товаров;
- управление категориями товаров:
 - добавление категорий;
 - удаление категорий;
 - редактирование категорий;
 - скрытие категорий;
- управление пользователями:
 - просмотр пользователей;
 - поиск пользователей;
 - изменение данных;
 - просмотр операций пользователей;
- управление заказами:
 - просмотр заказов;
 - редактирование заказов;
 - удаление заказов;
 - оплата заказов;
- переписка с пользователями.

4.1. Вход администратора

Вход для администратора ничем не отличается от входа для другого пользователя. Но в таблице `users` поле `type` для администратора имеет значение 9 (признак администратора). Это приведет к формированию для администратора других пунктов главного меню (рис. 4.1) и других наборов подпрограмм для каждого пункта

главного меню. Напоминаю, что эти наборы хранятся в таблице базы данных mainmenu.

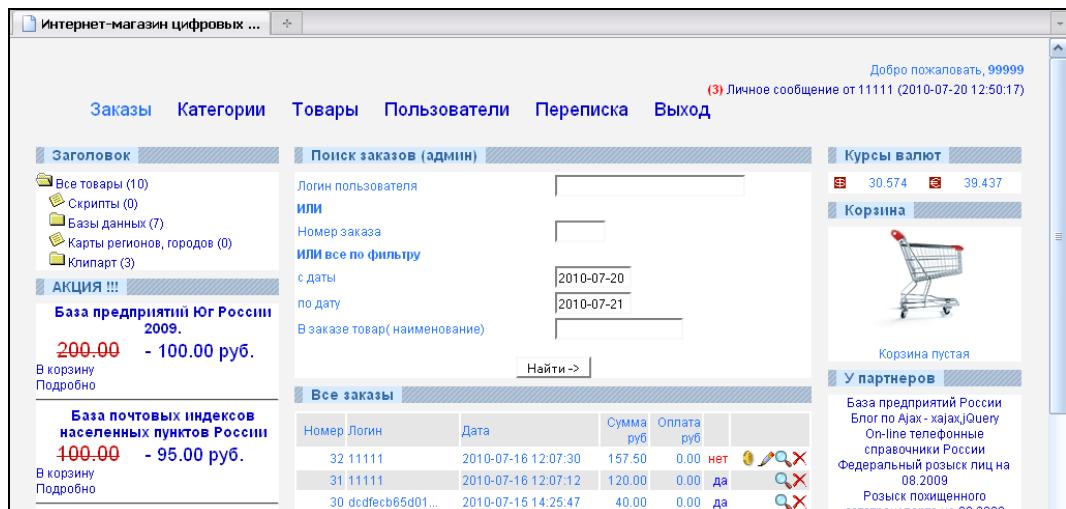


Рис. 4.1. Главное меню администратора

4.2. Управление товарами

При просмотре, поиске товаров, просмотре товара подробно для администратора вызываются те же функции формирования контента, что и для обычных пользователей, но есть различие: вывод дополнительных ссылок — **Редактировать** и **Добавить новый товар** (рис. 4.2), а при просмотре товара подробно еще и **Удалить** и **Скрыть** (или **Открыть**) (рис. 4.3).

Рис. 4.2. Ссылки **Редактировать** и **Добавить новый товар** для администратора

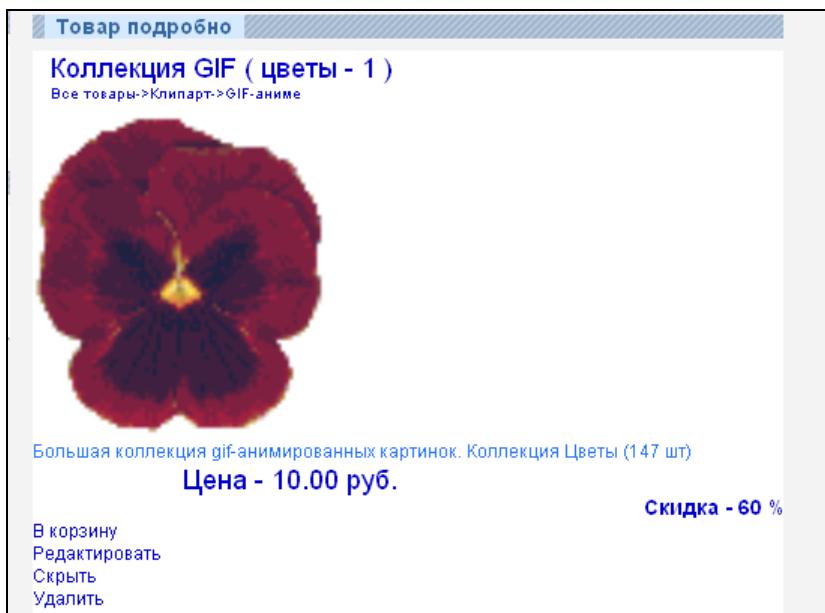


Рис. 4.3. Ссылки для администратора при просмотре товара подробно

4.2.1. Добавление нового товара

При нажатии на ссылку **Добавить новый товар** (см. рис. 4.2) из файла prgtovars_admin/add_new_tovar.php (листинг 4.1) вызывается хайп-функция Add_New_Tovar.

Листинг 4.1

```
<?php
// Добавление нового товара
function Add_New_Tovar ()
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // заголовок
    $zag=f_zag1("Добавление нового товара");
    // вывод заголовка
    $objResponse->assign("centercaption5","innerHTML",$zag);
    // формирование контента – формы нового товара
    $content=f_add_new_tovar();
    // вывод контента
    $objResponse->assign("center5","innerHTML",$content);
    // блок center5 в зону видимости
    $objResponse->script("document.getElementById
```

```
('center5').scrollIntoView();";  
$objResponse->assign("flag_ajax","value",'no');  
return $objResponse;  
}  
?  
?
```

Формирование контента — формы ввода для нового товара — выполняет функция `f_add_new_tovar`, расположенная в файле `prgtovars_admin/function_add_new_tovar.php` (листинг 4.2).

Листинг 4.2

```
<?php  
// Вывод формы регистрации товара  
function f_add_new_tovar()  
{  
    // подключение файла настроек  
    require_once("my.php");  
    // подключение к базе данных  
    require_once("mybaza.php");  
    $text1=<center>;  
    $text1.=<form id='FormRegTovar' action='javascript:void(null);'  
        onsubmit='xajax.\\"ButtonFormRegTovar\\").disabled=true;  
        xajax.\\"ButtonFormRegTovar\\").value=\"Подождите...\";  
        xajax_Go_Add_New_Tovar(xajax.getFormValues(  
            \"FormRegTovar\"));'>;  
    $text1.=<table width=100%>;  
    $text1.=<caption>Форма регистрации товара</caption>;  
    // дерево категорий  
    $text1.=<input type='text' name='id_tovar' id='id_tovar' value='0' >;  
    // выборка из базы родительского элемента  
    $query1="SELECT id FROM kategory WHERE id_parent='0'";  
    $rez1=mysql_query($query1);  
    $row1=mysql_fetch_row($rez1);  
    $rezfun=f_regtovar_open_kategory($row1[0]);  
    $text1.=<tr><td width=30% valign=top>Категория</td>  
        <td width=50%><center>  
            <input type='text' name='tovar_kategory' id='tovar_kategory'  
            value=' ' onchange='xajax_Control_Reg_Tovar  
                (xajax.getFormValues(\"FormRegTovar\"));'>;  
    $text1.=<div class='menu' style='margin-left:0' id=regtovar_kategory1>  
        ".$rezfun."</div>;  
    $text1.=</td>  
        <td width=20%>  
            <div id='reg_tovar_kategory'><font color='red'>no</font>
```

```
</div>
</td></tr>";
// картинка товара
$text1.=<tr><td width=30% valign=top>Картинка</td>
<td width=50%><center><img id=new_tovar_img
src='resize_100.php?pic=imgtovar/nofoto.gif'>
<br><input type='text' name=name_tovar_img id=name_tovar_img
value='imgtovar/nofoto.gif'>
</center>
<div id='iframe_img'><iframe src='upload_img.php?id=1'
frameborder='0'></iframe></div>
</td>
<td width=20%>
<div id='reg_tovar_img'><font color='blue'>ok</font>
</div>
</td></tr>";
// Нименование товара
$text1.=<tr><td width=30%>Название</td>
<td width=50%><input type='text' name='name' id='name'
size='30' maxlength='50' value='' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'></td>
<td width=20%>
<div id='reg_tovar_name'><font color='red'>no</font></div>
</td></tr>";
// Краткое описание
$text1.=<tr><td width=30%>Краткое описание</td>
<td width=50%><textarea name='info' id='info'
cols='30' rows='5' value='' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'></textarea>
</td>
<td width=20%>
<div id='reg_tovar_info'><font color='red'>no</font></div>
</td></tr>";
// Полное описание
$text1.=<tr><td width=30%>Полное описание</td>
<td width=50%><textarea name='fullinfo' id='fullinfo'
cols='30' rows='5' value='' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'></textarea>
</td>
<td width=20%>
<div id='reg_tovar_fullinfo'><font
color='red'>no</font></div>
```

```
</td></tr>";
// Цена
$text1.=<tr><td width=30%>Цена, руб.</td>
<td width=50%><input type='text' name='pay_rub' id='pay_rub'
size='10' maxlength='10' value='' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'></td>
<td width=20%>
<div id='reg_tovar_pay_rub'><font
color='red'>no</font></div>
</td></tr>";
// Цена по акции
$text1.=<tr><td width=30%>Цена (акция), руб.</td>
<td width=50%><input type='text' name='new_pay_rub'
id='new_pay_rub' size='10' maxlength='10' value='' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'></td>
<td width=20%>
<div id='reg_tovar_new_pay_rub'><font
color='red'>no</font></div>
</td></tr>";
// Сам цифровой товар - архив
$text1.=<tr><td width=30% valign=top>Архив</td>
<td width=50%><center><img id=new_arhiv_img
src='imgtovar/null.gif'>
<br><input type='text' name=name_tovar_arhiv
id=name_tovar_arhiv value=''
onchange='xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'>
<div id='iframe_arhiv'><iframe src='upload_arhiv.php?id=1'
frameborder='0'></iframe></div>
</center>
</td>
<td width=20%>
<div id='reg_tovar_arhiv'><font color='red'>no</font>
</div>
</td></tr>";
// Кнопка submit
$text1.=<tr><td></td>
<td>
<input type='submit' id='ButtonFormRegTovar'
value='Зарегистрировать ->' disabled=true>
</td>
<td></td></tr>";
```

```
$text1.=</table>;
$text1.=</form>;
return $text1;
}
?>
```

Вид формы регистрации нового товара приведен на рис. 4.4 и 4.5.

The screenshot shows a registration form for a new product. On the left, there's a sidebar with user information: 'Пользователь: Гость', 'Заказов - 32', 'Заказов оплаченных: 22', and a 'Контакты' section containing a green phone icon, the number '385771293', an email icon, 'my_shop@bk.ru', a telephone icon, '+79107855718', and a 'WMID 131672524818'. The main form area has several fields:

- 'Категория': A dropdown menu showing 'Все товары (10)', 'Скрипты (0)', 'Базы данных (7)', 'Карты регионов, городов (0)', and 'Клипарт (3)'. It includes a placeholder 'imgtovar/nofoto.gif' and a file upload button 'Обзор...'.
- 'Название': An input field with a red 'no' validation message.
- 'Краткое описание': A large text area with a red 'no' validation message.

Рис. 4.4. Форма регистрации нового товара

This screenshot shows the continuation of the product registration form. It contains the following fields:

- 'Краткое описание': A large text area with a red 'no' validation message.
- 'Полное описание': A large text area with a red 'no' validation message.
- 'Цена, руб.': An input field with a red 'no' validation message.
- 'Цена (акция), руб.': An input field with a red 'no' validation message.
- 'Архив': A file upload field with a placeholder 'Загрузите на сервер архив zip,rar,gz' and a 'Обзор...' button. It also has a red 'no' validation message.

Рис. 4.5. Форма регистрации нового товара (продолжение)

Рядом с каждым полем находится блок для отображения правильности заполнения этого поля. Изначально кнопка **Зарегистрировать** неактивна. Она станет активной после правильного заполнения всех полей. Правильность большинства полей (кроме полей загрузки картинки и файла архива) проверяет хайп-функция Control_Reg_Tovar, которая вызывается из файла prgtovars_admin/control_reg_tovar.php (листинг 4.3) по событию onchange. Ей передаются значения полей формы FormRegTovar.

Листинг 4.3

```
<?php
// Проверка правильности заполнения полей
// при регистрации товара
function Control_Reg_Tovar($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    $count=0; // счетчик правильно заполненных полей
    // подключение к базе данных
    require_once("mybaza.php");
    $id_kategory=explode(";",trim(utf8towin($Id[tovar_kategory])));
    $query1="SELECT id FROM kategory WHERE id='".$id_kategory[0]."' ";
    $rez1=mysql_query($query1);
    if(mysql_num_rows($rez1)>0)
    {
        $query2="SELECT COUNT(id) FROM kategory WHERE
            id_parent='".$id_kategory[0]."' ";
        $rez2=mysql_query($query2);
    }
    // категория
    if(trim(utf8towin($Id[tovar_kategory]))=='')
        $objResponse->assign("reg_tovar_kategory","innerHTML",<font
            color='red'>no</font>");

    elseif(mysql_num_rows($rez1)==0)
        $objResponse->assign("reg_tovar_kategory","innerHTML",<font
            color='red'>Категории с ID='".$id_kategory[0]."' не
            существует </font>");

    elseif(mysql_result($rez2,0)>0)
        $objResponse->assign("reg_tovar_kategory","innerHTML",<font
            color='red'>Товар заносится в категории, не имеющие
            вложений</font>");

    else
    {
        $objResponse->assign("reg_tovar_kategory","innerHTML",<font
            color='blue'>OK</font>");
    }
}
```

```
$count++;
}
// название
if(strlen(trim(utf8towin($Id[name])))>10)
{
    $objResponse->assign("reg_tovar_name","innerHTML","<font
        color='blue'>OK</font>");
$count++;
}
else
{
    $objResponse->assign("reg_tovar_name","innerHTML","<font
        color='red'>no</font>");
}
// краткое описание
if(strlen(trim(utf8towin($Id[info])))>20)
{
    $objResponse->assign("reg_tovar_info","innerHTML","<font
        color='blue'>OK</font>");
$count++;
}
else
{
    $objResponse->assign("reg_tovar_info","innerHTML","<font
        color='red'>no</font>");
}
// подробное описание
if(strlen(trim(utf8towin($Id[fullinfo])))>20)
{
    $objResponse->assign("reg_tovar_fullinfo","innerHTML","<font
        color='blue'>OK</font>");
$count++;
}
else
{
    $objResponse->assign("reg_tovar_fullinfo","innerHTML","<font
        color='red'>no</font>");
}
// цена
if(ereg("^( [0-9]{1}) ([0-9\.]{1,9})$",$Id[pay_rub]) &&
    substr_count($Id[pay_rub],".")<2)
{
    $objResponse->assign("reg_tovar_pay_rub","innerHTML","<font
        color='blue'>OK</font>");
$count++;
}
```

```

    }
else
    $objResponse->assign("reg_tovar_pay_rub","innerHTML","<font
        color='red'>ERR</font>");
// цена акция
if(ereg("^(0-9]{1})([0-9\.]{1,9})$",$Id[new_pay_rub]) &&
    substr_count($Id[new_pay_rub],".")<2)
{
    $objResponse->assign("reg_tovar_new_pay_rub","innerHTML","<font
        color='blue'>OK</font>");
    $count++;
}
else
    $objResponse->assign("reg_tovar_new_pay_rub","innerHTML","<font
        color='red'>ERR</font>");
// архив
if(file_exists($Id[name_tovar_arhiv]))
{
    $objResponse->assign("reg_tovar_arhiv","innerHTML","<font
        color='blue'>OK</font>");
    $count++;
}
else
    $objResponse->assign("reg_tovar_arhiv","innerHTML","<font
        color='red'>ERR</font>");
// проверка, все ли поля правильно заполнены
if($count==7)
    $objResponse->assign("ButtonFormRegTovar","disabled",false);
else
    $objResponse->assign("ButtonFormRegTovar","disabled",true);
$objResponse->assign("flag_ajax","value",'no');
return $objResponse();
}
?>
```

Файлы товара и картинки подгружаются на сервер без перезагрузки страницы. Загрузка осуществляется в момент выбора файла. Для выбора картинки и файла товара в форму встроены фреймы, в которых есть форма загрузки файла:

```
<div id='iframe_img'><iframe src='upload_img.php?id=1'
frameborder='0'></iframe></div>
<div id='iframe_arhiv'><iframe src='upload_arhiv.php?id=1'
frameborder='0'></iframe></div>
```

В момент выбора файла по событию запускаем функцию `upload_img` во фрейме и загружаем файл на сервер, а из фрейма в форму передаем только имя уже загруженного файла (рис. 4.6). Это результат выполнения скриптов `upload_img.php`

и upload_arhiv.php, содержимое которых приведено в листинге 4.4 и 4.5 соответственно.

Листинг 4.4

```
<?php
if ($_FILES["image"]["error"] == 0)
{
;
$ftmp = $_FILES['image']['tmp_name'];
$oname = $_FILES['image']['name'];
$fname = "imgtovar/".$time()."";;
$pictype="";
switch(strtolower($_FILES['image']['type']))
{
    case "image/jpeg" : $pictype=".jpg"; break;
    case "image/pjpeg": $pictype=".jpg"; break;
    case "image/gif"  : $pictype=".gif"; break;
    case "image/png"  : $pictype=".png"; break;
    case "image/x-png": $pictype=".png"; break;
    default           : $pictype="";   break;
}
if($pictype=="")
{ ;?>
<?
if($_GET[id]!=1){ ?>
<html><head><script>
var par = window.parent.document;
par.forms.FormRegTovar.elements.name_tovar_img.value=
"imgtovar/nofoto.gif";
var new_tovar_img = par.getElementById('new_tovar_img');
<? if($_GET[id]==2) {?>
    new_tovar_img.src = 'img/zagruzka.gif';
<? }
else {?>
    new_tovar_img.src = 'resize_100.php?pic=imgtovar/nofoto.gif';
<? } ?>
par.getElementById('reg_tovar_img').innerHTML=<font color='red'>He
заргужено</font>";
<? }
else { ;?>
<html><head><script>
var par = window.parent.document;
var new_tovar_img = par.getElementById('new_tovar_img');
var old_img = par.getElementById('name_tovar_img').value;
```

```
new_tovar_img.src = 'resize_100.php?pic='+old_img;
;
<? }
}
else
{
$fname = "imgtovar/".$time().$pictype;
move_uploaded_file($ftmp, $fname);
chmod($fname,0777);
?>
<html><head><script>
var par = window.parent.document;
var new_tovar_img = par.getElementById('new_tovar_img');
new_tovar_img.src = 'resize_100.php?pic=<? echo $fname; ?>';
par.forms.FormRegTovar.elements.name_tovar_img.value=<? echo $fname; ?>;
par.getElementById('reg_tovar_img').innerHTML=<font color='blue'>
Зарегистрировано</font>";
<?php
}

}
else
{ ;?><html><head>
<script language="javascript">
<? } ?>
function upload_img()
{
var par = window.parent.document;
var new_tovar_img = par.getElementById('new_tovar_img');
new_tovar_img.src = 'img/zagruzka.gif';
document.iform.submit();
}
</script>
<style>
iframe {
    border-width: 0px;
    height: avto;
    width: 200px;
}
iframe.hidden {
    visibility: hidden;
    width:0px;
    height:0px;
}
#file {
```

```
width: 150px;  
}  
</style>  
</head><body><center>  
<form name="iform" action="" method="post" enctype="multipart/form-data">  
Загрузите на сервер фото <br> *.jpg, *.png, *.gif<br>  
<input id="file" type="file" name="image" onchange="upload_img()" />  
</form>  
</center>  
</body>  
</html>
```

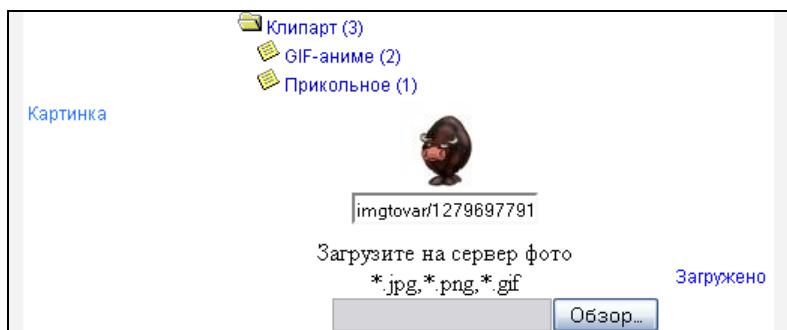


Рис. 4.6. Загрузка файла на сервер без перезагрузки страницы

Листинг 4.5

```
<?php  
if ($_FILES["arhiv"]["error"] == 0)  
{  
    $ftmp = $_FILES['arhiv']['tmp_name'];  
    $oname = $_FILES['arhiv']['name'];  
    $fname = "arhivtovar/" . time() . "";  
    $type="";  
    if(strtolower($_FILES['arhiv']['type'])=="application/x-gzip-  
        compressed")  
        $type=".gz";  
    elseif(strtolower($_FILES['arhiv']['type'])=="application/x-gzip")  
        $type=".gz";  
    elseif(strtolower($_FILES['arhiv']['type'])=="application/x-zip-  
        compressed")  
        $type=".zip";  
    elseif(strtolower($_FILES['arhiv']['type'])=="application/x-rar-  
        compressed")
```

```
$type=".rar";
elseif(strtolower($_FILES['arhiv']['type'])=="application/octet-stream"
    && substr($_FILES['arhiv']['name'],
        strpos($_FILES['arhiv']['name'], "."),
        strlen($_FILES['arhiv']['name'])-1)=='.rar')
$type=".rar";
else
$type="";

if($type!="")
{$fname = "arhivtovar/".$time().$type;
move_uploaded_file($ftmp, $fname);
chmod($fname,0777);
?>
<html><head><script>
var par = window.parent.document;
var new_arhiv_img = par.getElementById('new_arhiv_img');
new_arhiv_img.src = 'img/null.gif';
par.forms.FormRegTovar.elements.name_tovar_arhiv.value="
<? echo $fname; ?>";
par.getElementById('reg_tovar_arhiv').innerHTML=<font
color='blue'>ok</font>;
<?}
else {
if($_GET[id] != 1 ){?>
<html><head><script>
var par = window.parent.document;
var new_arhiv_img = par.getElementById('new_arhiv_img');
new_arhiv_img.src = 'img/null.gif';
par.forms.FormRegTovar.elements.name_tovar_arhiv.value="";
par.getElementById('reg_tovar_arhiv').innerHTML=<font color='red'>He
заружено</font>;
<?php }
else { ?>
<html><head><script>
var par = window.parent.document;
var new_arhiv_img = par.getElementById('new_arhiv_img');
new_arhiv_img.src = 'img/null.gif';
<?php }
}
}
else
{?><html><head>
<script language="javascript">
<? } ?>
```

```
function upload_arhiv()
{
    var par = window.parent.document;
    var new_arhiv_img = par.getElementById('new_arhiv_img');
    new_arhiv_img.src = 'img/zagruzka.gif';
    document.iform.submit();
</script>
<style>
iframe {
    border-width: 0px;
    height: avto;
    width: 200px;
}
iframe.hidden {
    visibility: hidden;
    width:0px;
    height:0px;
}
#file {
    width: 150px;
}
</style>
<head><body><center>
<form name="iform" action="" method="post" enctype="multipart/form-data">
Загрузите на сервер <br>архив zip,rar,gz<br>
<input id="file" type="file" name="arhiv" onchange="upload_arhiv()" />
</form>
</center>
</body>
</html>
```

Если все поля заполнены правильно, становится активной кнопка **Зарегистрировать**. При ее нажатии из файла prgtovars_admin/go_add_new_tovar.php (листинг 4.6) вызывается хайп-функция Go_Add_New_Tovar, которой передаются все значения формы FormRegTovar. Функция записывает данные нового товара в базу и выводит сообщение об успешном добавлении товара (рис. 4.7). При этом статус товара устанавливается скрытым.

Листинг 4.6

```
<?php
// Регистрация нового товара
// Сохранение изменений отредактированного товара
function Go_Add_New_Tovar($Id)
{
    $objResponse = new xajaxResponse();
```

```
$objResponse->assign("flag_ajax","value",'yes');
require_once("my.php");
require_once("mybaza.php");

$id_tovar=$Id[id_tovar];
$arr_id_kategory=explode(";", $Id[tovar_kategory]);
$id_kategory=$arr_id_kategory[0];
$name=utf8towin($Id[name]);
$info=utf8towin($Id[info]);
$fullinfo=utf8towin($Id[fullinfo]);
$pay_rub=$Id[pay_rub];
$new_pay_rub=$Id[new_pay_rub];
$img=$Id[name_tovar_img];
$arhiv=$Id[name_tovar_arhiv];
$data=date('Y-m-d H:i:s');
// добавление нового товара
if($id_tovar>0)
{
    $query0="SELECT visible FROM tovars WHERE id='".$id_tovar."'";
    $visible=mysql_result(mysql_query($query0),0);
    $query1="UPDATE tovars SET
        id_kategory='".$id_kategory."',name='".$name."',
        info='".$info."',fullinfo='".$fullinfo."',
        pay_rub='".$pay_rub."',new_pay_rub='".$new_pay_rub."',
        img='".$img."',arhiv='".$arhiv."',
        data_update='".$data."',visible='no'
        WHERE id='".$id_tovar."'";
    $rez1=mysql_query($query1);
}
// изменение отредактированного товара
else
{
    $visible='no';
    $query1="INSERT INTO tovars SET
        id_kategory='".$id_kategory."',name='".$name."',
        info='".$info."',fullinfo='".$fullinfo."',
        pay_rub='".$pay_rub."',new_pay_rub='".$new_pay_rub."',
        img='".$img."',arhiv='".$arhiv."',
        data='".$data."',visible='no'
        ";
    $rez1=mysql_query($query1);
    $id_tovar=mysql_insert_id();
}
if(!$rez1)
    $text1=<center>Ошибка mysql (новый товар) </center>".$query1;
else
```

```

$text1=<center><br>Успешно! <br><br>
    <a href='javascript:void()' onclick='
        xajax_View_Tovar(\".$id_tovar.\");'>
        Товар подробно</a></center>";
if($Id[id_tovar]>0 && $visible=='yes')
{
    $id_parent=$id_kategory;
    while($id_parent>0)
    {
        $query3="SELECT id,id_parent,nn FROM kategory WHERE
        id='".$id_parent."'";
        $rez3=mysql_query($query3);
        $id_parent=mysql_result($rez3,0,"id_parent");
        $id_kategory=mysql_result($rez3,0,"id");
        $nn=mysql_result($rez3,0,"nn");
        $new_nn=$nn-1;
        $query4="UPDATE kategory SET nn='".$new_nn."' WHERE
        id='".$id_kategory."'";
        mysql_query($query4);
    }
    // изменить вид блока категорий
    // т.к. изменилось количество видимых
    $content=f_open_kategory(1);
    $objResponse->assign("kategory1","innerHTML",$content);
}
// вывод контента
$objResponse->assign("center5","innerHTML",$text1);
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

ЗАМЕЧАНИЕ

Функция Go_Add_New_Tovar используется не только для добавления в базу данных нового товара, но и для внесения изменений при редактировании товара.

По ссылке **Товар подробно** (см. рис. 4.7) можно перейти на подробный просмотр содержимого товара.



Рис. 4.7. Успешное добавление нового товара

4.2.2. Редактирование товара

При нажатии на ссылку **Редактировать** (см. рис. 4.2) вызывается хайп-функция `Edit_Tovar`, расположенная в файле `prgtovars_admin/edit_tovar.php` (листинг 4.7).

Листинг 4.7

```
<?php
// Редактирование товара
function Edit_Tovar($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // заголовок
    $zag=f_zag1("Добавление нового товара");
    // вывод заголовка
    $objResponse->assign("centercaption5","innerHTML",$zag);
    // формирование контента - формы редактирования товара
    $content=f_edit_tovar($Id);
    // вывод контента
    $objResponse->assign("center5","innerHTML",$content);
    // блок center5 в зону видимости
    $objResponse->script("document.getElementById
        ('center5').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Формирование контента — формы ввода для нового товара — выполняет функция `f_edit_tovar`, расположенная в файле `prgtovars_admin/function_edit_tovar.php` (листинг 4.8). Вид формы редактирования товара иллюстрирует рис. 4.8.

Листинг 4.8

```
<?php
// Вывод формы редактирования товара
// $Id id товара
function f_edit_tovar($Id)
{
    require_once("mybaza.php");
    require_once("my.php");
    // Получение данных товара
    $query0="SELECT * FROM tovars WHERE id='".$Id."'";
    $rez0=mysql_query($query0);
    $row0=mysql_fetch_assoc($rez0);
    $text1=<center>;
```

```
$text1.=<form id='FormRegTovar' action='javascript:void(null);'  
    onsubmit='  
        xajax.$(\"ButtonFormRegTovar\").disabled=true;  
        xajax.$(\"ButtonFormRegTovar\").value=\"Подождите...\";  
        xajax_Go_Add_New_Tovar(xajax.getFormValues(  
            \"FormRegTovar\"));'>;  
  
$text1.=<table width=100%>;  
$text1.=<caption>Форма редактирования товара</caption>;  
$text1.=<input type='text' name='id_tovar' id='id_tovar'  
    value='".$Id."' >;  
// выборка из базы родительского элемента  
$query1="SELECT id FROM kategory WHERE id_parent='0'";  
$rez1=mysql_query($query1);  
$row1=mysql_fetch_row($rez1);  
$rezfun=f_regtovar_open_kategory($row1[0]);  
$query2="SELECT name FROM kategory WHERE id='".$row0[id_kategory]."';  
$rez2=mysql_query($query2);  
$kategory=$row0[id_kategory].";".mysql_result($rez2,0);  
$text1.=<tr><td width=30% valign=top>Категория</td>  
    <td width=50%><center>  
        <input type='text' name='tovar_kategory' id='tovar_kategory'  
            value='".$kategory."' onchange='xajax_Control_Reg_Tovar  
                (xajax.getFormValues(\"FormRegTovar\"));'>;  
$text1.=<div class='menu' style='margin-left:0'  
    id='regtovar_kategory1'>".$rezfun."</div>;  
$text1.=</td>  
    <td width=20%>  
        <div id='reg_tovar_kategory'><font color='blue'>ok</font>  
        </div>  
    </td></tr>;  
// картинка  
$text1.=<tr><td width=30% valign=top>Картинка</td>  
    <td width=50%><center><img id=new_tovar_img  
        src='resize_100.php?pic='".$row0[img]."'>  
        <br><input type='text' name=name_tovar_img id=name_tovar_img  
            value='".$row0[img]."'>  
        </center>  
        <div id='iframe_img'><iframe src='upload_img.php?id=1'  
            frameborder='0'></iframe></div>  
    </td>  
    <td width=20%>  
        <div id='reg_tovar_img'><font color='blue'>ok</font>  
        </div>  
    </td></tr>;  
// наименование  
$text1.=<tr><td width=30%>Название</td>  
    <td width=50%><input type='text' name='name' id='name'
```

```
size='30' maxlength='50' value='".$row0[name]."' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'></td>
<td width=20%>
<div id='reg_tovar_name'><font color='blue'>ok</font></div>
</td></tr>";
// краткое описание
$text1.=<tr><td width=30%>Краткое описание</td>
<td width=50%><textarea name='info' id='info'
cols='30' rows='5' value='".$row0[info]."' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'>
".$row0[info]."</textarea>
</td>
<td width=20%>
<div id='reg_tovar_info'><font color='blue'>ok</font></div>
</td></tr>";
// полное описание
$text1.=<tr><td width=30%>Полное описание</td>
<td width=50%><textarea name='fullinfo' id='fullinfo'
cols='30' rows='5' value='".$row0[fullinfo]."' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'>
".$row0[fullinfo]."</textarea>
</td>
<td width=20%>
<div id='reg_tovar_fullinfo'><font color='blue'>
ok</font></div>
</td></tr>";
// цена
$text1.=<tr><td width=30%>Цена, руб.</td>
<td width=50%><input type='text' name='pay_rub' id='pay_rub'
size='10' maxlength='10' value='".$row0[pay_rub]."'>
onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'></td>
<td width=20%>
<div id='reg_tovar_pay_rub'><font color='blue'>
ok</font></div>
</td></tr>";
// цена акция
$text1.=<tr><td width=30%>Цена (акция), руб.</td>
<td width=50%><input type='text' name='new_pay_rub'
id='new_pay_rub' size='10' maxlength='10'
value='".$row0[new_pay_rub]."' onchange='
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'></td>
```

```

<td width=20%>
    <div id='reg_tovar_new_pay_rub'><font color='blue'>
        ok</font></div>
    </td></tr>";
// архив
$text1.=<"tr><td width=30% valign=top>Архив</td>
<td width=50%><center><img id=new_arhiv_img
src='imgtovar/null.gif'>
<br><input type='text' name=name_tovar_arhiv
id=name_tovar_arhiv value='".$row0[arhiv]."'
onchange='xajax_Control_Reg_Tovar(
xajax.getFormValues(\"FormRegTovar\"));'>
<div id='iframe_arhiv'><iframe src='upload_arhiv.php?id=1'
frameborder='0'></iframe></div>
</center>
</td>
<td width=20%>
    <div id='reg_tovar_arhiv'><font color='blue'>ok</font>
    </div>
</td></tr>";
$text1.=<"tr><td></td>
<td><input type='submit' id='ButtonFormRegTovar'
value='Изменить ->'></td>
<td></td></tr>";
$text1.=</table>";
$text1.=</form>";
return $text1;
}
?>

```

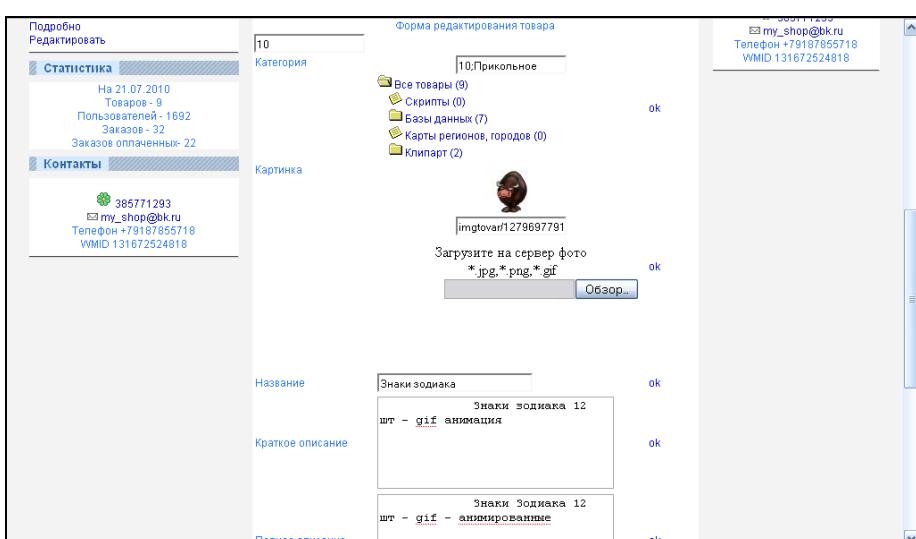


Рис. 4.8. Форма редактирования товара

При изменении данных идет проверка корректности заполнения полей и в случае неверного редактирования поля кнопка **Изменить** становится неактивной. При нажатии на кнопку **Изменить** происходит запись изменений товара. Товар после редактирования становится скрытым.

Для выбора (изменения) категории товара при его создании (редактировании) выводится дерево категорий (рис. 4.9).

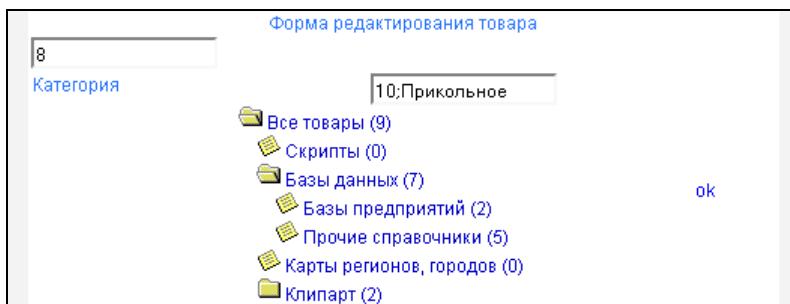


Рис. 4.9. Дерево категорий при создании или редактировании товара

Хайк-функции для работы с деревом категорий — `RegTovar_Open_Kategory` и `RegTovar_Close_Kategory` — находятся в файле `prgtovars_admin/regtovar_open_close_category.php` (листинг 4.9).

Листинг 4.9

```
<?php
// раскрытие категории
function RegTovar_Open_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // получить контент
    $content=f_regtovar_open_kategory($Id);
    $objResponse->assign("regtovar_kategory".$Id,"innerHTML",$content);
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}

// сворачивание категории
function RegTovar_Close_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // получить контент
    $content=f_regtovar_close_kategory($Id);
    $objResponse->assign("regtovar_kategory".$Id,"innerHTML",$content);
```

```
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

Категории товара формируют функции `f_regtovar_open_kategory.php` и `f_regtovar_close_kategory.php`, расположенные в файле `prgtovars_admin/function_regtovar_kategory.php` (листинг 4.10). Они подобны функциям открытия/закрытия категорий, отличие — при выборе нижней категории данные о ней (`id` и `name`) передаются в поле формы создания (редактирования) товара.

Листинг 4.10

```
<?php
/// Выдача дерева категорий
//Раскрытие каталога - показать вложенные подкаталоги
function f_regtovar_open_kategory($Id)
{
    require_once("mybaza.php");
    $text1="";
    // получение списка вложенных категорий
    $query1="SELECT name,id,nn FROM kategory WHERE id='".$Id."'"
        && visible='yes' ";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    $text1.="
```

```

<img src='img/open_dir.ico'></a></span>
<span><a href='javascript:void();' onclick='
xajax_RegTovar_Open_Kategory(\".$row2[id].\")'
\".$row2[name].\" (\".$row2[nn].\")</a><span></div>";
}
else
{
$text1.= "<div class='menu' id='regtovar_kategory\".$row2[id].\"'
<a href='javascript:void();' onclick='
document.forms.FormRegTovar.tovar_kategory.value=
\"\".\".$row2[id].\";\".$row2[name].\"\";
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'>
<img src='img/last_dir.ico'></a>
<a href='javascript:void();' onclick='
document.forms.FormRegTovar.tovar_kategory.value=
\"\".\".$row2[id].\";\".$row2[name].\"\";
xajax_Control_Reg_Tovar(xajax.getFormValues(
\"FormRegTovar\"));'>
\".$row2[name].\" (\".$row2[nn].\")</a></div>";
}
return $text1;
}

//*****Закрытие каталога *****
function f_regtovar_close_kategory($Id)
{
require_once("mybaza.php");
$text1="";
// получение списка вложенных категорий
$query1="SELECT name,id,nn FROM kategory WHERE id=\"".$Id.""
&& visible='yes' ";
$rez1=mysql_query($query1);
$row1=mysql_fetch_assoc($rez1);
$text1.= "<a href='javascript:void();' onclick='
xajax_RegTovar_Open_Kategory(\".$Id.\")'
<img src='img/open_dir.ico'></a>
<a href='javascript:void();' onclick='
xajax_RegTovar_Open_Kategory(\".$Id.\")'
\".$row1[name].\" (\".$row1[nn].\")</a>";
return $text1;
}
?>
```

4.2.3. Удаление товара

Для удаления товара необходимо при подробном просмотре товара нажать ссылку **Удалить** (рис. 4.10). При этом вызывается хайп-функция `Delete_Tovar`, расположенная в файле `prgtovars_admin/delete_tovar.php` (листинг 4.11). Из базы данных товар не удаляется, функция `Delete_Tovar` устанавливает в таблице `tovars` для записи данного товара значение поля `visible=del`. Обычные пользователи не смогут видеть этот товар, но, если не истек срок действия ссылки, его по-прежнему можно скачать.

Листинг 4.11

```
<?php
// Удалить товар (установить visible='del')
function Delete_Tovar($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключиться к базе данных
    require_once("mybaza.php");
    $query2="SELECT id_kategori,visible FROM tovars WHERE id='".$Id."'";
    $visible=mysql_result(mysql_query($query2),0,"visible");
    // установить значение поля visible=del
    $query1="UPDATE tovars SET visible='del' WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    // пересчитать товар в категориях, если было visible='yes'
    if($visible=='yes')
    {
        $id_parent=mysql_result(mysql_query($query2),0,"id_kategori");
        while($id_parent>0)
        {
            $query3="SELECT id,id_parent,nn FROM kategory WHERE
                    id='".$id_parent."'";
            $rez3=mysql_query($query3);
            $id_parent=mysql_result($rez3,0,"id_parent");
            $id_kategori=mysql_result($rez3,0,"id");
            $nn=mysql_result($rez3,0,"nn");
            $new_nn=$nn-1;
            $query4="UPDATE kategory SET nn='".$new_nn."' WHERE
                    id='".$id_kategori."'";
            mysql_query($query4);
        }
        // изменить вид категорий
        $content=f_open_kategori(1);
        $objResponse->assign("kategori1","innerHTML",$content);
```

```

        }
// показать этот товар со статусом удален
$content=f_view_tovar($Id);
$objResponse->assign("center5","innerHTML",$content);
$zagcontent=f_zag1("Товар подробно");
$objResponse->assign("centercaption5","innerHTML",$zagcontent);
$objResponse->script("document.getElementById
    ('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

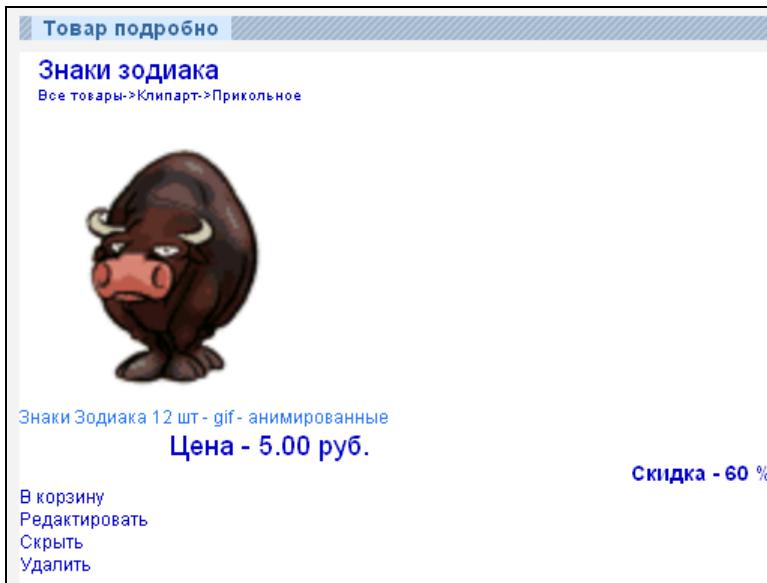


Рис. 4.10. Ссылки Удалить и Скрыть для товара

4.2.4. Скрытие товара, открытие товара

Иногда возникает необходимость временно скрыть товар от показа, не удаляя его, а затем снова открыть. В таблице `tovars` базы данных значение поля `visible=no` соответствует скрытию, а `visible=yes` открытию товара. Для реализации этого созданы функции скрытия/открытия товара — `DoHidden_Tovar` и `DoVisible_Tovar`, расположенные соответственно в файлах `prgtovars_admin/dohidden_tovar.php` (листинг 4.12) и `prgtovars_admin/dovisible_tovar.php` (листинг 4.13). При скрытии/открытии товаров не забываем пересчитывать количество товаров в категориях.

Листинг 4.12

```
<?php
// Скрыть товар
function DoHidden_Tovar ($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // установить visible=no
    $query1="UPDATE tovars SET visible='no' WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    // пересчитать товар в категориях
    $query2="SELECT id_kategory FROM tovars WHERE id='".$Id."'";
    $id_parent=mysql_result(mysql_query($query2),0);
    while($id_parent>0)
    {
        $query3="SELECT id,id_parent,nn FROM kategory WHERE
            id='".$id_parent."'";
        $rez3=mysql_query($query3);
        $id_parent=mysql_result($rez3,0,"id_parent");
        $id_kategory=mysql_result($rez3,0,"id");
        $nn=mysql_result($rez3,0,"nn");
        $new_nn=$nn-1;
        $query4="UPDATE kategory SET nn='".$new_nn."' WHERE
            id='".$id_kategory."'";
        mysql_query($query4);
    }
    // показать измененные категории (количество)
    $content=f_open_kategory(1);
    $objResponse->assign("kategory1","innerHTML",$content);
    // вывести товар
    $content=f_view_tovar($Id);
    $objResponse->assign("center5","innerHTML",$content);
    $zagcontent=f_zag1("Товар подробно");
    $objResponse->assign("centercaption5","innerHTML",$zagcontent);
    $objResponse->script("document.getElementById
        ('center5').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Листинг 4.13

```
<?php
// Открыть товар
function DoVisible_Tovar($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключиться к базе данных
    require_once("mybaza.php");
    // установить visible=no
    $query1="UPDATE tovars SET visible='yes' WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    // пересчитать товар в категориях
    $query2="SELECT id_kategory FROM tovars WHERE id='".$Id."'";
    $id_parent=mysql_result(mysql_query($query2),0);
    while($id_parent>0)
    {
        $query3="SELECT id,id_parent,nn FROM kategory WHERE
            id='".$id_parent."'";
        $rez3=mysql_query($query3);
        $id_parent=mysql_result($rez3,0,"id_parent");
        $id_kategory=mysql_result($rez3,0,"id");
        $nn=mysql_result($rez3,0,"nn");
        $new_nn=$nn+1;
        $query4="UPDATE kategory SET nn='".$new_nn."' WHERE
            id='".$id_kategory."'";
        mysql_query($query4);
    }
    // показать измененные категории (количество)
    $content=f_open_kategory(1);
    $objResponse->assign("kategory1","innerHTML",$content);
    // показать товар
    $content=f_view_tovar($Id);
    $objResponse->assign("center5","innerHTML",$content);
    $zagcontent=f_zag1("Товар подробно");
    $objResponse->assign("centercaption5","innerHTML",$zagcontent);
    $objResponse->script("document.getElementById
        ('center5').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

При входе администратора в блок **Товары** открывается список скрытых товаров (рис. 4.11).

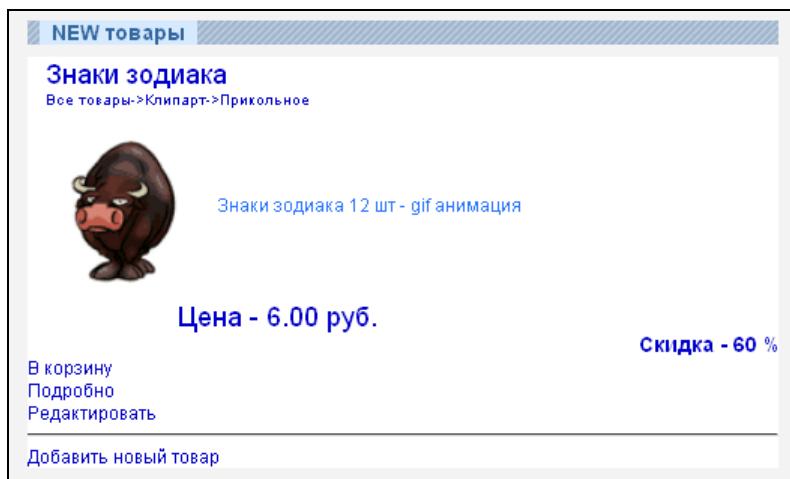


Рис. 4.11. Список скрытых товаров при входе

4.3. Управление категориями товаров

Управление категориями товаров включает:

- добавление;
- удаление;
- редактирование;
- скрытие.

Для управления категориями товаров нажимаем на ссылку **Категории** в главном меню администраторского профиля. Вид формы для управления категориями приведен на рис. 4.12.

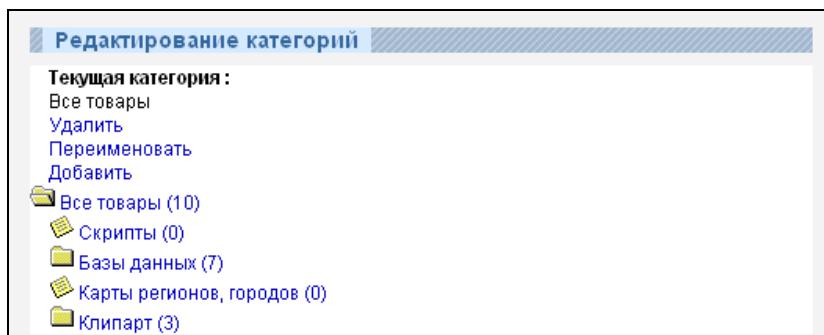


Рис. 4.12. Форма редактирования категорий

Функция Admin_Tek_Kategory, показывающая путь к выбранной категории, и функции Admin_Open_Kategory и Admin_Close_Kategory, отвечающие за раскрытие и закрытие категорий, расположены в файле prgkategory/admin_open_close_category.php (листинг 4.14).

Листинг 4.14

```
<?php
// раскрытие категории
function Admin_Open_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // формирование контента
    $content=f_admin_open_kategory($Id);
    // выдача контента - дерева категорий
    $objResponse->assign("admin_kategory".$Id,"innerHTML",$content);
    // формирование пути к текущей категории
    // и ссылок
    $content=f_string_kategory($Id);
    $content=<b>Текущая категория :</b><br>".$content;
    $content.= "<br><a href='javascript:void();' onclick='
        xajax_Delete_Admin_Kategory(\".$Id.\");'>Удалить</a>";
    $content.= "<br><a href='javascript:void();' onclick='
        xajax_Rename_Admin_Kategory(\".$Id.\");'>Переименовать</a>";
    $content.= "<br><a href='javascript:void();' onclick='
        xajax_Add_Admin_Kategory(\".$Id.\");'>Добавить</a>";
    // выдача контента - пути и ссылок
    $objResponse->assign("admin_path_kategory","innerHTML",$content);
    $objResponse->script("document.getElementById
        ('admin_path_kategory').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
// скрытие категории
function Admin_Close_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // формирование контента
    $content=f_admin_close_kategory($Id);
    // выдача контента - дерева категорий
    $objResponse->assign("admin_kategory".$Id,"innerHTML",$content);
    // формирование пути к текущей категории
    // и ссылок
```

```

$content=f_string_kategory($Id);
$content=<b>Текущая категория :</b><br>".$content;
$content.=<br><a href='javascript:void();' onclick=
    xajax_Delete_Admin_Kategory(".$Id.");'>Удалить</a>";
$content.=<br><a href='javascript:void();' onclick=
    xajax_Rename_Admin_Kategory(".$Id.");'>Переименовать</a>";
$content.=<br><a href='javascript:void();' onclick=
    xajax_Add_Admin_Kategory(".$Id.");'>Добавить</a>";
// выдача контента - пути и ссылок
$objResponse->assign("admin_path_kategory","innerHTML",$content);
$objResponse->script ("document.getElementById
    ('admin_path_kategory').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
// путь к текущей категории
function Admin_Tek_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // получение пути текущей категории
    $content=f_string_kategory($Id);
    $content=<b>Текущая категория :</b><br>".$content;
    $content.=<br><a href='javascript:void();' onclick=
        xajax_Delete_Admin_Kategory(".$Id.");'>Удалить</a>";
    $content.=<br><a href='javascript:void();' onclick=
        xajax_Rename_Admin_Kategory(".$Id.");'>Переименовать</a>";
    $content.=<br><a href='javascript:void();' onclick=
        xajax_Add_Admin_Kategory(".$Id.");'>Добавить</a>";
    // выдача контента
    $objResponse->assign("admin_path_kategory","innerHTML",$content);
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>

```

Функции формирования контента дерева категорий `f_admin_open_kategory` и `f_admin_close_kategory` находятся в файле `prgkategory/function_admin_open_close_category.php` (листинг 4.15).

Листинг 4.15

```

<?php
/// Выдача дерева категорий
//*****Раскрытие каталога *****
// показать вложенные подкаталоги

```

```
function f_admin_open_kategory($Id)
{
    require_once("mybaza.php");
    $text1="";
    // получение списка вложенных категорий
    $query1="SELECT name,id,nn FROM kategory WHERE id='".$Id."'"
        && visible='yes' ";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    $text1.=""
                <span><a href='javascript:void\(\);' onclick='
                    xajax\_Admin\_Open\_Kategory\(\".\$row2\[id\].\"\)'
                    <img src='img/open\_dir.ico'></a></span>
                    <span><a href='javascript:void\(\);' onclick='
                    xajax\_Admin\_Open\_Kategory\(\".\$row2\[id\].\"\)'
                    \"\$row2\[name\].\" \(\".\$row2\[nn\].\"\)</a><span></div>";
        }
        else
        {
            \$text1.="
```

```

function f_admin_close_kategory($Id)
{
    require_once("mybaza.php");
    $text1="";
    // получение списка вложенных категорий
    $query1="SELECT name,id,nn FROM kategory WHERE id='".$Id."'"
        && visible='yes' ";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    $text1.="

```

4.3.1. Добавление категорий товаров

Для добавления категории товара сначала необходимо выбрать текущую категорию, куда будет добавляться новая. Это выполняют, переходя по дереву категорий и щелкнув мышью по нужной категории. Затем следует нажать на ссылку **Добавить**. Из файла prgkategory/add_admin_kategory.php (листинг 4.16) вызывается хайп-функция Add_Admin_Kategory, которая выдает форму добавления категории (рис. 4.13).

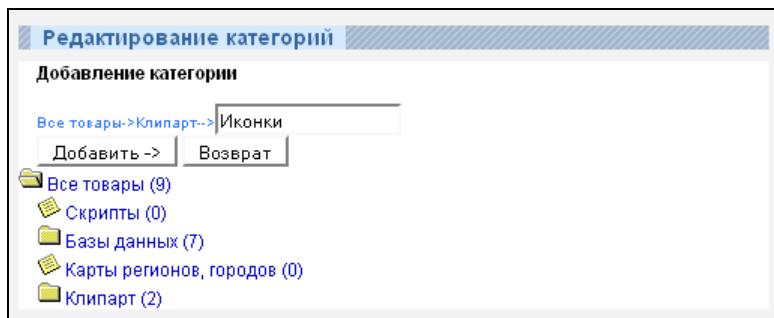


Рис. 4.13. Форма добавления категории

Листинг 4.16

```

<?php
// Форма добавления новой категории
function Add_Admin_Kategory($Id)
{

```

```

$objResponse = new xajaxResponse();
$objResponse->assign("flag_ajax","value",'yes');
// подключаемся к базе данных
require_once("mybaza.php");
$text1=<b>Добавление категории</b>";
$text1.=<form id='FormAdminKategory' action='javascript:void(null);'
    onsubmit='xajax.\\"ButtonFormAdminKategory\\").disabled=true;
    xajax.\\"ButtonFormAdminKategory\\").value=
    \\\"Подождите...\\";
    xajax_Go_Add_Admin_Kategory(xajax.getFormValues(
        \\"FormAdminKategory\\"));\\>";
$text1.=<input type='hidden' name='id_parent' value='\".$Id.\"' >";
$text1.=<br>.f_string_kategory($Id).--><input type='text'
    name='name' value=' ' ;
$text1.=<br><input type='submit' id='ButtonFormAdminKategory'
    value='Добавить ->'> ";
$query1="SELECT id FROM kategory WHERE id_parent='\".$Id.\"' &&
    visible='yes' ";
$rez1=mysql_query($query1);
if(mysql_num_rows($rez1)>0)
    $text1.=<input type='button' value='Возврат' onclick='
        xajax_Admin_Open_Kategory(\".$Id.\");\\></form\\>";
else
    $text1.=<input type='button' value='Возврат' onclick='
        xajax_Admin_Tek_Kategory(\".$Id.\");\\></form\\>";
$objResponse->assign("admin_path_kategory","innerHTML",$text1);
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```



Рис. 4.14. Добавлена категория Иконки

В поле вводим название категории и нажимаем на кнопку **Добавить**. Данные формы передаются хайпа-функции Go_Add_Admin_Kategory, которая добавляет новую категорию товаров в базу (родительской категорией для нее будет текущая категория). Новая категория добавится в дерево категорий (рис. 4.14). Функция Go_Add_Admin_Kategory находится в файле prgkategery/go_add_admin_category.php (листинг 4.17).

Листинг 4.17

```
<?php
// Добавление новой категории
function Go_Add_Admin_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // проверка переданного значения
    if(strlen(utf8towin($Id[name]))==0)
    {
        $objResponse->alert("Пустое название !!!");
        $objResponse->assign("flag_ajax","value",'no');
        return $objResponse;
    }
    // добавление
    $query1="INSERT INTO kategory SET id_parent='".$Id[id_parent]."',
        name='".$utf8towin($Id[name])."',nn='0',visible='yes' ";
    $rez1=mysql_query($query1);
    if(!$rez1)
    {
        $objResponse->alert("Ошибка добавления !!!");
        $objResponse->assign("flag_ajax","value",'no');
        return $objResponse;
    }
    // заново переоткрыть родительскую категорию
    $parent=$Id[id_parent];
    $content=f_admin_open_kategory($parent);
    $objResponse->assign("admin_kategory".$parent,"innerHTML",$content);
    $content=f_string_kategory($parent);
    $content=<b>Текущая категория :</b><br>".$content;
    $content.= "<br><a href='javascript:void();' onclick='
        xajax_Delete_Admin_Kategory(\".$parent.\");'>Удалить</a>";
    $content.= "<br><a href='javascript:void();' onclick='
        xajax_Rename_Admin_Kategory(\".$parent.\");'>
        Переименовать</a>";
```

```
$content.= "<br><a href='javascript:void();' onclick='
    xajax_Add_Admin_Kategory(\".$parent.\");'>Добавить</a>";
$objResponse->assign("admin_path_kategory", "innerHTML", $content);
$objResponse->script ("document.getElementById
    ('admin_path_kategory').scrollIntoView();");
$objResponse->assign("flag_ajax", "value", 'no');
return $objResponse;
}
```

?>

4.3.2. Редактирование категорий товаров

Для редактирования категории товара сначала необходимо выбрать текущую категорию. Затем следует нажать на ссылку **Переименовать**. Из файла prgkategory/rename_admin_category.php (листинг 4.18) вызывается хайп-функция Rename_Admin_Kategory, которая выдает форму добавления категории (рис. 4.15).

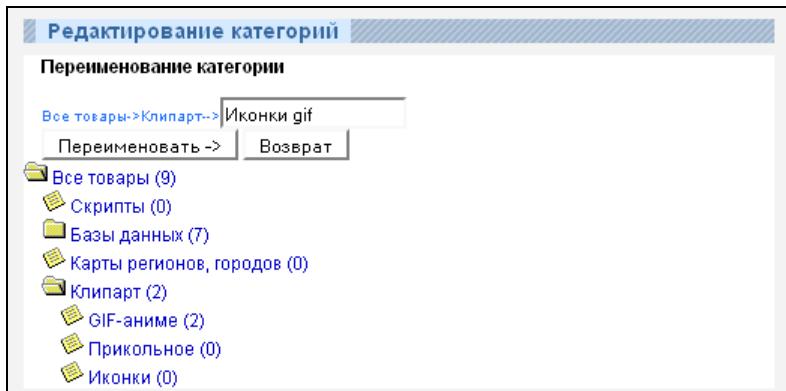


Рис. 4.15. Форма переименования категории

Листинг 4.18

```
<?php
// Форма переименования категории
function Rename_Admin_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax", "value", 'yes');
    // подключение к базе
    require_once("mybaza.php");
    $text1=<b>Переименование категории</b>";
    $text1.= "<form id='FormAdminCategory' action='javascript:void(null);'>
```

```

onsubmit='xajax.$(\"ButtonFormAdminCategory\").disabled=true;
xajax.$(\"ButtonFormAdminCategory\").value=
\"Подождите...\";
xajax_Go_Rename_Admin_Kategory(xajax.getFormValues(
    \"FormAdminKategory\"));'>;
$text1.= "<input type='hidden' name='id_kategory' value='".$Id."' ";
$text1.= "<br>";
// путь к категориям
$content=f_string_category($Id);
$content=substr($content,0,strrpos($content,"->"));
$query0="SELECT name FROM kategory WHERE id='".$Id."'";
$rez0=mysql_query($query0);
$name=mysql_result($rez0,0);
$text1.=$content."--><input type='text' name='name'
    value='".$name."' >";
$text1.= "<br><input type='submit' id='ButtonFormAdminCategory'
    value='Переименовать ->' >";
$query1="SELECT id FROM kategory WHERE id_parent='".$Id."' &&
    visible='yes' ";
$rez1=mysql_query($query1);
if(mysql_num_rows($rez1)>0)
    $text1.= " <input type='button' value='Возврат' onclick='
        xajax_Admin_Open_Kategory(\".$Id.\");'></form>";
else
    $text1.= " <input type='button' value='Возврат' onclick='
        xajax_Admin_Tek_Kategory(\".$Id.\");'></form>";
$objResponse->assign("admin_path_kategory","innerHTML",$text1);
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

В поле вводим название категории и нажимаем на кнопку **Переименовать**. Данные формы передаются хаях-функции `Go_Rename_Admin_Kategory`, которая добавляет новую категорию товаров в базу (родительской категорией для нее будет текущая категория). В дереве категорий видим измененную категорию (рис. 4.16). Функция `Go_Rename_Admin_Kategory` находится в файле `prgkategory/go_rename_admin_category.php` (листинг 4.19).

Листинг 4.19

```

<?php
// Добавление новой категории
function Go_Rename_Admin_Kategory($Id)
{

```

```
$objResponse = new xajaxResponse();
$objResponse->assign("flag_ajax","value",'yes');
// подключение к базе данных
require_once("mybaza.php");
// проверка
if(strlen(utf8towin($Id[name]))==0)
{
    $objResponse->alert("Пустое название !!!");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
// переименование
$query1="UPDATE kategory SET name='".utf8towin($Id[name])."'
        WHERE id='".$Id[id_kategory]."' ";
$rez1=mysql_query($query1);
if(!$rez1)
{
    $objResponse->alert("Ошибка переименования !!!");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
// заново переоткрыть категорию
$query2="SELECT id_parent FROM kategory WHERE
        id='".$Id[id_kategory]."' ";
$rez2=mysql_query($query2);
$parent=mysql_result($rez2,0);
$content=f_admin_open_kategory($parent);
$objResponse->assign("admin_kategory".$parent,"innerHTML",$content);
$content=f_string_kategory($parent);
$content=<b>Текущая категория :</b><br>.$content;
$content.= "<br><a href='javascript:void();' onclick='
            xajax_Delete_Admin_Kategory(\".$parent.\");'>Удалить</a>";
$content.= "<br><a href='javascript:void();' onclick='
            xajax_Rename_Admin_Kategory(\".$parent.\");'>
            Переименовать</a>";
$content.= "<br><a href='javascript:void();' onclick='
            xajax_Add_Admin_Kategory(\".$parent.\");'>Добавить</a>";
$objResponse->assign("admin_path_kategory","innerHTML",$content);
$objResponse->script("document.getElementById
                    ('admin_path_kategory').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

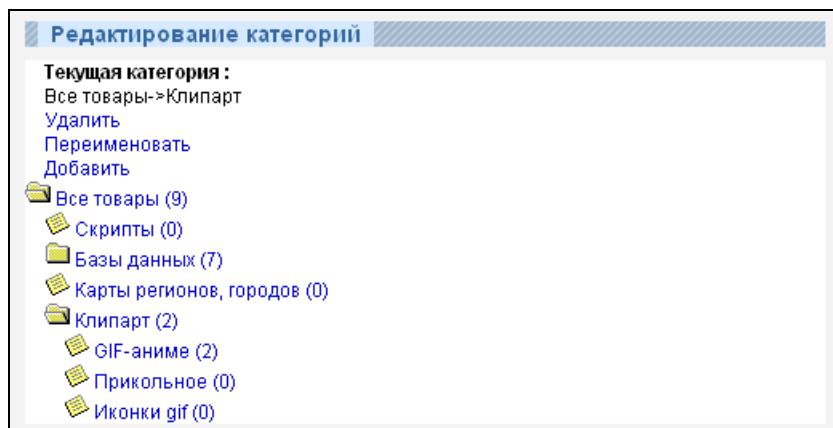


Рис. 4.16. Категория Иконки переименована в Иконки gif

4.3.3. Удаление категорий товаров

Для удаления категории товаров сначала необходимо выбрать текущую категорию. Затем следует нажать на ссылку Удалить. Из файла prgkategory/delete_admin_kategory.php (листинг 4.20) вызывается хайник-функция Delete_Admin_Kategory. При попытке удалить категорию, в которой находятся товары, функция выдаст предупреждение **Нельзя удалять непустые категории** (рис. 4.17). При попытке удалить категорию, имеющую вложенные категории, функция выдаст предупреждение **Нельзя удалять категории, имеющие вложения** (рис. 4.18). Из базы данных категория не удаляется, устанавливается значение поля visible=no, и она становится невидимой (рис. 4.19).

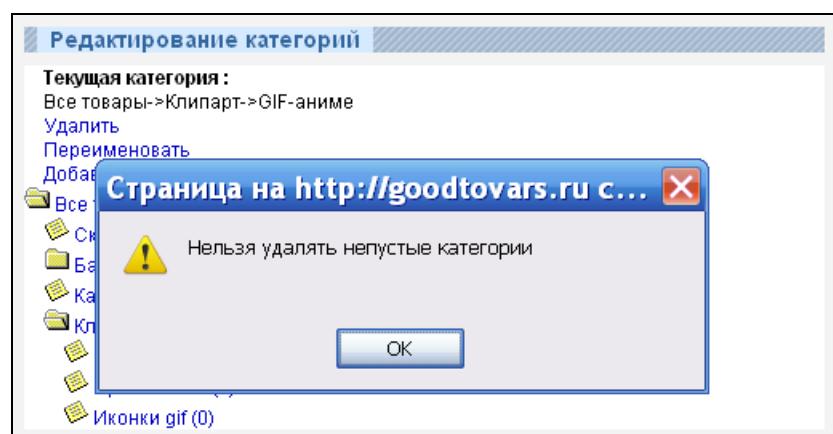


Рис. 4.17. Предупреждение Нельзя удалять непустые категории

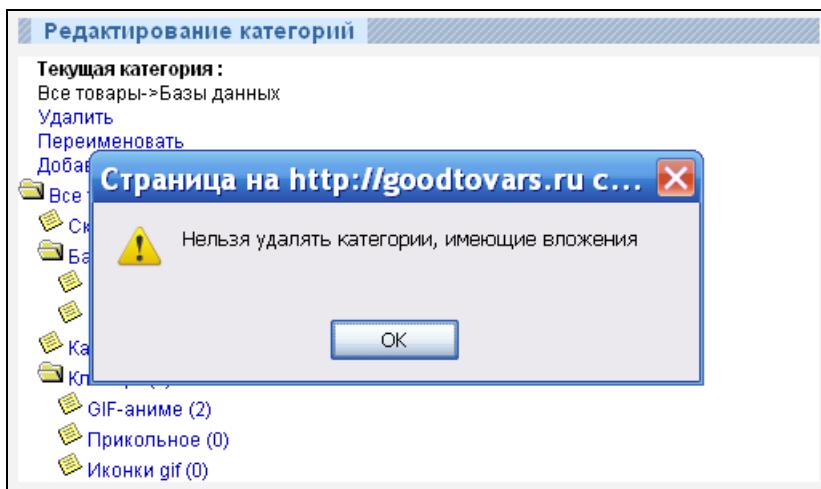


Рис. 4.18. Предупреждение Нельзя удалять категории, имеющие вложения

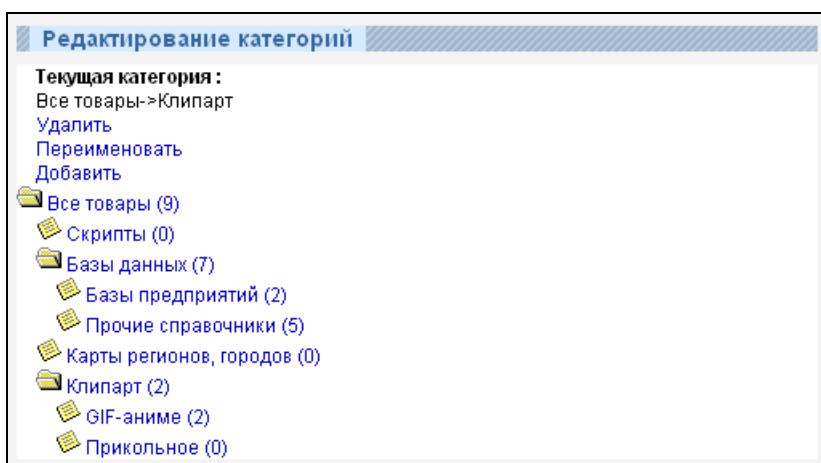


Рис. 4.19. Была удалена категория Иконки gif

Листинг 4.20

```
<?php
// Удаление категории
function Delete_Admin_Kategory($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // проверка 1 - удалять категории, имеющие вложения нельзя !!!
    $query1="SELECT id FROM kategory WHERE id_parent='".$Id."'";
}
```

```
&& visible='yes' ";
$rez1=mysql_query($query1);
if(mysql_num_rows($rez1)>0)
{
    $objResponse->alert("Нельзя удалять категории,
имеющие вложения ");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}

// проверка 2 - удалять непустые нельзя !!!
$query2="SELECT nn FROM kategory WHERE id='".$Id."'";
$rez2=mysql_query($query2);
if(mysql_result($rez2,0)>0)
{
    $objResponse->alert("Нельзя удалять непустые категории ");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}

// "удаление" - установка visible=no
$query3="UPDATE kategory SET visible='no' WHERE id='".$Id."'";
$rez3=mysql_query($query3);
if(!$rez3)
{
    $objResponse->alert("Ошибка удаления !!!");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}

// заново переоткрыть родительскую категорию
$query4="SELECT id_parent FROM kategory WHERE id='".$Id."'";
$rez4=mysql_query($query4);
$parent=mysql_result($rez4,0);
$content=f_admin_open_kategory($parent);
$objResponse->assign("admin_kategory".$parent,"innerHTML",$content);
$content=f_string_kategory($parent);
$content."<b>Текущая категория :</b><br>".$content;
$content.= "<br><a href='javascript:void();' onclick='
    xajax_Delete_Admin_Kategory(\".$parent.\");'>Удалить</a>";
$content.= "<br><a href='javascript:void();' onclick='
    xajax_Rename_Admin_Kategory(\".$parent.\");'>
    Переименовать</a>";
$content.= "<br><a href='javascript:void();' onclick='
    xajax_Add_Admin_Kategory(\".$parent.\");'>Добавить</a>";
$objResponse->assign("admin_path_kategory","innerHTML",$content);
$objResponse->script("document.getElementById
    ('admin_path_kategory').scrollIntoView();");
```

```
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>
```

4.4. Управление заказами

Администратору необходимо просматривать все заказы пользователей, значит, должна быть возможность корректировать заказы, вплоть до изменения цены, "оплачивать" заказы пользователей — переводить заказ в статус "Оплачен". Заказов может быть много, поэтому нужен хороший фильтр поиска, чтобы найти заказ по номеру, вывести заказы конкретного пользователя или за определенный период, отыскать заказы с определенным товаром.

4.4.1. Просмотр заказов пользователей

При выборе из профиля администратора пункта главного меню **Заказы** из файла `prgzakaz_admin/view_aal_zakaz_admin.php` (листинг 4.21) вызывается хайник-функция `View_All_Zakaz_Admin`, которая выдает список заказов всех пользователей постранично (рис. 4.20).

Все заказы						
Номер	Логин	Дата	Сумма руб	Оплата руб		
32 11111		2010-07-16 12:07:30	157.50	0.00	нет	  
31 11111		2010-07-16 12:07:12	120.00	0.00	да	 
30 dcdfecb65d01...		2010-07-15 14:25:47	40.00	0.00	да	 
25 182ea8d02e1f...		2009-11-25 17:18:33	10.00	0.00	да	 
24 de97a1b8a6b3...		2009-11-21 20:45:40	95.00	0.00	да	 
23 190025270e2d...		2009-11-05 17:52:20	10.00	0.00	да	 
22 b255c8db7e3a...		2009-11-05 15:08:59	95.00	0.00	да	
21 b255c8db7e3a...		2009-11-05 14:52:09	45.00	0.00	да	
20 b255c8db7e3a...		2009-11-05 14:32:17	10.00	0.00	да	
19 99999		2009-11-05 14:32:06	80.00	0.00	да	
1 2 3 >>						
Всего - 27 Страниц - 3						

Рис. 4.20. Список заказов всех пользователей постранично

Листинг 4.21

```
<?php
// Просмотр всех заказов пользователя постранично (админ)
// $Id - номер страницы
```

```
function View_All_Zakaz_Admin($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // получение контента
    $content=f_view_all_zakaz_admin($Id);
    // вывод контента
    $objResponse->assign("center3","innerHTML",$content[0]);
    // вывод навигатора страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
    // блок center3 в зону видимости
    $objResponse->script("document.getElementById
        ('center3').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
```

?>

Контент для вывода заказов всех пользователей формирует функция `f_view_all_zakaz_admin`. Для каждого заказа в списке выводим следующую информацию:

- номер заказа;
- логин пользователя, сделавшего заказ;
- дата и время заказа;
- сумма заказа, руб.;
- сумма поступившей оплаты, руб.;
- статус оплаты заказа:
 - да;
 - нет;
- ссылки:
 - оплатить (для неоплаченных);
 - редактировать (для неоплаченных);
 - посмотреть;
 - удалить.

Формирование контента для вывода списка заказов происходит в функции `f_view_all_zakaz`, которая находится в файле `prgzakaz_admin/function_view_all_zakaz.php` (листинг 4.22). Число заказов на страницу — константа `NN2` в файле `my.php`.

Листинг 4.22

```
<?php
// Просмотр заказов пользователя постранично (дата по убыванию) (админ)
```

```
// $Id - номер страницы для показа
function f_view_all_zakaz_admin($Id)
{
// подключение файла настроек
require_once("my.php");
// подключение к базе данных
require_once("mybaza.php");
$text=array();
$text1="";
$query0="SELECT COUNT(id) FROM zakaz WHERE visible='yes'
        ORDER BY data DESC ";
$rez0=mysql_query($query0);
$count=mysql_result($rez0,0);
$pages=ceil($count/NN2);
$page=min($Id,$pages);$poz=($page-1)*NN2;
$text1.=<div class='zag_view_tovars'>";
if($count>0)
{
$query1="SELECT * FROM zakaz WHERE visible='yes'
        ORDER BY data DESC LIMIT ".$poz.", ".NN2."'";
$rez1=mysql_query($query1);
// шапка для таблицы
$text1.=<table>;
$text1.=<tr><td class='str0' align=right>Номер</td>";
$text1.=<td class='str0'>Логин</td>";
$text1.=<td class='str0'>Дата</td>";
$text1.=<td class='str0' align=right>Сумма<br>руб</td>";
$text1.=<td class='str0' align=right>Оплата<br>руб</td>";
$text1.=<td class='str0'></td>";
$text1.=<td class='str0'></td></tr>";$i=0;
while($row1=mysql_fetch_assoc($rez1))
{$i++;
// номер заказа
$text1.=<tr><td class='str' .($i%2+1). "'"
    align=right>".$row1[id]."</td>";
$query3="SELECT login FROM users WHERE id='".$row1[id_user]."' ";
$rez3=mysql_query($query3);
$login=mysql_result($rez3,0);
// логин сократить
$login=(strlen($login)<13)?($login):(substr($login,0,12)."...");
// логин
$text1.=<td class='str' .($i%2+1). "'>".$login."</td>";
// дата
$text1.=<td class='str' .($i%2+1). "'>".$row1[data]."</td>";
// сумма заказа
```

```
$text1.= "<td class='str'." . ($i%2+1) . "'  
        align=right>".$row1[summa_rub]."</td>";  
// сумма оплаченная  
$text1.= "<td class='str'." . ($i%2+1) . "'  
        align=right>".$row1[summa_rub_oplata]."</td>";  
// статус оплаты  
if($row1[pay]=='yes')  
    $text1.= "<td class='str'." . ($i%2+1) . "' align=right><font  
            color='blue'>да</font></td>";  
else  
    $text1.= "<td class='str'." . ($i%2+1) . "' align=right><font  
            color='red'>нет</font></td>";  
$text1.= "<td class='str'." . ($i%2+1) . "' align=right>";  
// ссылки  
if($row1[pay]=='no')  
{  
    $text1.= "<a href='javascript:void();' onclick='  
        xajax_Oplata_Zakaz_Admin(\".$row1[id]\");'  
        title='Оплатить'><img src='img/pay.gif'></a>";  
    $text1.= "<a href='javascript:void();' onclick='  
        xajax_Edit_Zakaz_Admin(\".$row1[id]\");'  
        title='Редактировать'><img src='img/edit.png'></a>";  
}  
$text1.= "<a href='javascript:void();' onclick='  
        xajax_View_Zakaz_Admin(\".$row1[id]\");'  
        title='Подробно'><img src='img/view.gif'></a>";  
$text1.= "<a href='javascript:void();' onclick='  
        xajax_Delete_Zakaz_Admin(\".$row1[id]\");'  
        title='Удалить'><img src='img/delete.png'></a>";  
$text1.= "</td></tr>";  
}  
$text1.= "</table>";  
// список ссылок перехода по страницам  
$text2="";  
if($pages>1)  
{  
    if($page != 1)  
    {$i=$page-1;  
    $text2.= "<a href='javascript:void(null);' onclick='  
        var x=new Array();x=".($page-1).";  
        xajax_View_All_Zakaz_Admin(x);'> <<<</a>";  
    }  
    $x=array();  
    $x=doarray1($page,$pages,5);  
    for($i=0;$i < count($x);$i++)
```

```

//for($i=1;$i <= $pages;$i++)
{
if($x[$i]==$page)
    $text2."<a href='javascript:void(null);' onclick='
        var x=new Array();x=". $x[$i].";
        xajax_View_All_Zakaz_Admin(x);'> ".$x[$i]."</a>";
else
{
    $text2."<a href='javascript:void(null);' onclick='
        var x=new Array();x=". ($page+1).";
        xajax_View_All_Zakaz_Admin(x);'>></a>";
}
if($page != $pages)
{$i=$page+1;
$text2."<a href='javascript:void(null);' onclick='
        var x=new Array();x=". ($page+1).";
        xajax_View_All_Zakaz_Admin(x);'>></a>";
}
if($pages != 1)
{$text2.= "<br><br>Всего - ".$count." Страниц - ".$pages."<br> </center>";}
else
{$text2.= "</center>";}
}
}
else
{
$text2."<br><center>Заказов не обнаружено</center><br>";
}
$text[0]=$text1;
$text[1]=$text2;
return $text;
}
?>

```

4.4.2. Просмотр заказов пользователей по фильтру

Вид формы поиска заказа по фильтру приведен на рис. 4.21. Поиск заказа осуществляется по:

- номеру заказа;
- логину пользователя (соответствие);
- периоду времени;
- наименованию товара в заказе (соответствие).

Форму поиска заказов генерирует функция `f_form_search_zakaz_admin`, расположенная в файле `prgzakaz_admin/function_form_search_zakaz_admin` (листинг 4.23).

Рис. 4.21. Форма поиска заказов по фильтру

Листинг 4.23

```
<?php
// Форма поиска заказа (admin)
function f_form_search_zakaz_admin()
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    // создание формы
    $text1.= "<form id='FormSearchZakaz' action='javascript:void(null);'
        onsubmit='xajax.\\"ButtonFormSearchZakaz\\").disabled=true;
        xajax.\\"ButtonFormSearchZakaz\\").value=\\\"Подождите...\\";
        xajax_View_Search_Zakaz_Admin(xajax.getFormValues(
        \\"FormSearchZakaz\\"));\\'>";
    $text1.= "<table width=100%>";
    // логин
    $text1.= "<tr><td width=50%>Логин пользователя</td></tr>";
    $text1.= "<td width=50%
        <input type='text' name='user_zakaz' value='' size=32
        maxlength=32>
        <input type='hidden' id='pagesearch' name='pagesearch'
        value='1'></td></tr>";
    $text1.= "<tr><td width=50%><b>ИЛИ</b></td></td></tr>";
    // номер заказа
    $text1.= "<tr><td width=50%>Номер заказа</td></tr>";
    $text1.= "<td width=50%
        <input type='text' name='number_zakaz' value='' size=5
        maxlength=5></td></tr>";
    $text1.= "<tr><td width=50%><b>ИЛИ все по фильтру</b></td></td></tr>";
```

```

// с даты
$text1.= "<tr><td width=50%>с даты</td>";
$text1.= "<td width=50%>
    <input type='text' name='datazakaz1' id='datazakaz1'
        value='".$date('Y-m-d')."' size=10 maxlength=10></td></tr>";
// по дату
$text1.= "<tr><td width=50%>по дату </td>";
$text1.= "<td width=50%>
    <input type='text' name='datazakaz2' id='datazakaz2'
        value='".$date('Y-m-d', strtotime('now +1 day'))."' size=10
        maxlength=10></td></tr>";
// наименование товара
$text1.= "<tr><td width=50%>В заказе товар( наименование)</td>";
$text1.= "<td width=50%>
    <input type='text' name='name_tovar' value='' size=20
        maxlength=20></td></tr>";
$text1.= "</table>";
$text1.= "<center><br><input type='submit' id='ButtonFormSearchZakaz'
    value='Найти ->'></center>";
$text1.= "</form>";
return $text1;
}
?>

```

После выбора параметров поиска и нажатия кнопки **Найти**, из файла prgza-kaz_admin/view_search_zakaz_admin.php (листинг 4.24) вызывается хайп-функция View_Search_Zakaz_Admin, которой в качестве параметров передаются значения формы FormSearchZakaz. Результат выводится в виде списка постранично.

Листинг 4.24

```

<?php
// Просмотр заказов по поиску (админ)
function View_Search_Zakaz_Admin($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // заголовок
    $zag=f_zag1("Результат поиска");
    $objResponse->assign("centercaption3","innerHTML",$zag);
    // формирование контента
    $content=f_view_search_zakaz_admin($Id);
    // вывод контента
    $objResponse->assign("center3","innerHTML",$content[0]);
    // вывод навигатора страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
}

```

```

// center3 в зону видимости
$objResponse->script("document.getElementById
    ('center3').scrollIntoView();");

// активировать кнопку submit
$objResponse->assign("ButtonFormSearchZakaz","value", "Найти ->");
$objResponse->assign("ButtonFormSearchZakaz","disabled", false);
$objResponse->assign("flag_ajax","value",'no');

return $objResponse;
}

?>

```

Выборку заказов по параметрам поиска и формирование контента для вывода осуществляет функция `f_view_search_zakaz_admin`, расположенная в файле `prgza-kaz_admin/function_view_search_zakaz_admin.php` (листинг 4.25). Пример результата поиска иллюстрирует рис. 4.22.

The screenshot shows a web-based administrative interface for managing orders. On the left, there's a sidebar with links for 'All products (9)', 'Scripts (0)', 'Data bases (7)', 'Regions and cities (0)', 'Clipboard (2)', and a promotional section for 'Business entities of Russia' with a 20% discount. Below that are sections for 'Post office index databases' and 'Federal travel operator directory' both offering 50% discounts. At the bottom of the sidebar is a 'Statistics' link. The main content area has tabs for 'Search orders (admin)', 'Courses of currency', 'Cart', 'Partners', and 'Contacts'. The 'Search orders (admin)' tab is active, displaying a search form with fields for 'Login' (empty), 'Order number' (empty), 'Or all by filter' (empty), 'From date' (07-01-2009), 'To date' (07-23-2010), and 'Product name' (base). A 'Search' button is present. Below the form is a table titled 'Search results' with columns: Order number, Login, Date, Sum, Payment, Status, and Actions. The table contains 11 rows of order data. To the right of the table are sections for 'Courses of currency' (USD: 30.406, EUR: 39.215) and 'Cart' (empty). The 'Partners' section lists various services and resources. The 'Contacts' section provides contact information for the shop.

Номер	Логин	Дата	Сумма	Оплата	руб	руб	Статус	Actions
9	2f4b910a580d...	2009-10-20 15:06:16	160.00	0.00		нет	да	
10	037472754c5a...	2009-10-21 07:28:00	100.00	100.00		да	да	
11	95e20bc42f0e...	2009-10-25 17:42:24	465.00	0.00		нет	да	
12	037472754c5a...	2009-10-25 20:39:07	210.00	210.00		да	да	
13	a83cca7e8eab...	2009-10-27 00:32:10	95.00	0.00		нет	да	
16	b255c8db7e3a...	2009-11-05 10:56:15	210.00	0.00		да	да	
19	99999	2009-11-05 14:32:06	80.00	0.00		да	да	
22	b255c8db7e3a...	2009-11-05 15:08:59	95.00	0.00		да	да	
24	de97a1b8a6b3...	2009-11-21 20:45:40	95.00	0.00		да	да	
31	11111	2010-07-16 12:07:12	120.00	0.00		да	да	

Рис. 4.22. Результат поиска заказов по фильтру

Листинг 4.25

```

<?php
// Просмотр заказов по поиску
// $Id - параметры формы
function f_view_search_zakaz_admin($Id)
{
    // подключение файла настроек
    require_once ("my.php");
}

```

```
// подключение к базе данных
require_once("mybaza.php");
$text=array();
$text1="";
// составление запроса
$query0="SELECT DISTINCT(zakaz.id) FROM zakaz,zakaz_table,tovars WHERE
zakaz.visible='yes' ";
$name_tovar=utftownin($Id[name_tovar]);
if(strlen(trim($Id[user_zakaz]))>0)
{
    $query0="SELECT DISTINCT(zakaz.id)FROM zakaz,zakaz_table,tovars,users
WHERE zakaz.visible='yes' ";
    $query0.="&& zakaz.id_user=users.id && LOWER(users.login) LIKE
'%".$Id[user_zakaz]."%' ";
}
elseif($Id[number_zakaz]>0)
{
    $query0.="&& zakaz.id='".$Id[number_zakaz]."' ";
}
else
{
    $query0.="&& zakaz.data >= '".$Id[datazakaz1]."' &&
zakaz.data <='".$Id[datazakaz2]."' ";
    if(strlen(rtrim(ltrim($name_tovar)))>0)
    {
        $query0.="&& LOWER(tovars.name) LIKE '%".$name_tovar."%' &&
zakaz_table.id_tovar=tovars.id
&& zakaz_table.id_zakaz=zakaz.id ";
    }
}
$rez0=mysql_query($query0);
$count=mysql_num_rows($rez0);
$pages=ceil($count/NN2);
$page=min($Id[pagesearch],$pages);$poz=($page-1)*NN2;
$text1=<div class='zag_view_tovars'>;
if($count>0)
{
    $query0.=" LIMIT ".$poz.", ".NN2."'";
    $query1=$query0;
    $rez1=mysql_query($query1);
    // шапка таблицы
    $text1.= "<table>";
    $text1.= "<tr><td class='str0' align=right>Номер</td>";
    $text1.= "<td class='str0'>Логин</td>";
    $text1.= "<td class='str0'>Дата</td>";
}
```

```
$text1.=<td class='str0' align=right>Сумма<br>руб</td>;
$text1.=<td class='str0' align=right>Оплата<br>руб</td>;
$text1.=<td class='str0'></td>;
$text1.=<td class='str0'></td></tr>;$i=0;
while($row1=mysql_fetch_row($rez1))
{
$i++;
$query2="SELECT * FROM zakaz WHERE id='".$row1[0]."' ";
$rez2=mysql_query($query2);
$row2=mysql_fetch_assoc($rez2);
$text1.= "<tr><td class='str'." . ($i % 2 + 1) . "'"
        . align=right>".$row2[id]."</td>";
$query3="SELECT login FROM users WHERE id='".$row2[id_user]."' ";
$rez3=mysql_query($query3);
$login=mysql_result($rez3,0);
// логин
$login=(strlen($login)<13)?($login):(substr($login,0,12)."....");
$text1.= "<td class='str'." . ($i % 2 + 1) . "'>".$login."</td>";
// дата
$text1.= "<td class='str'." . ($i % 2 + 1) . "'>".$row2[data]."</td>";
// сумма заказа
$text1.= "<td class='str'." . ($i % 2 + 1) . "'"
        . align=right>".$row2[summa_rub]."</td>";
// сумма оплаченная
$text1.= "<td class='str'." . ($i % 2 + 1) . "'"
        . align=right>".$row2[summa_rub_oplata]."</td>";
// статус оплаты
if($row2[pay]=='yes')
{
$text1.= "<td class='str'." . ($i % 2 + 1) . "' align=right><font"
        . color='blue'>да</font></td>";
}
else
{
$text1.= "<td class='str'." . ($i % 2 + 1) . "' align=right><font"
        . color='red'>нет</font></td>";
}
$text1.= "<td class='str'." . ($i % 2 + 1) . "' align=right>";
// ссылки
if($row2[pay]=='no')
{
$text1.= "<a href='javascript:void();' onclick='"
        . xajax_Oplata_Zakaz_Admin(".$row2[id].");'
        . title='Оплатить'><img src='img/pay.gif'></a>";
$text1.= "<a href='javascript:void();' onclick='"
        . xajax_Edit_Zakaz_Admin(".$row2[id].");'
        . title='Редактировать'><img src='img/edit.png'></a>";
}
$text1.= "<a href='javascript:void();' onclick='"
        . xajax_View_Zakaz_Admin(".$row2[id].");'
```

```
        title='Подробно'><img src='img/view.gif'></a>";
$text1.=<a href='javascript:void();' onclick=
    xajax_Delete_Zakaz_Admin("$.row2[id].";
    title='Удалить'><img src='img/delete.png'></a>";
$text1.= "</td></tr>";
}
$text1.=</table>;
// список ссылок перехода по страницам
$text2="";
if($pages>1)
{
if($page != 1)
{$i=$page-1;
$text2.=<a href='javascript:void(null);' onclick=
    document.forms.FormSearchZakaz.pagesearch.value=". $i .";
    xajax_View_Search_Zakaz_Admin(xajax.getFormValues(
    \"FormSearchZakaz\"));
    '> <<</a>";
}
$x=array();
$x=doarray1($page,$pages,5);
for($i=0;$i < count($x);$i++)
//for($i=1;$i <= $pages;$i++)
{
if($x[$i]==$page)
    $text2.=<a href=""> ".$x[$i]."</a>";
else
{
$text2.=<a href='javascript:void(null);' onclick=
    document.forms.FormSearchZakaz.pagesearch.value=". $x[$i] .";
    xajax_View_Search_Zakaz_Admin(xajax.getFormValues(
    \"FormSearchZakaz\"));
    '> ".$x[$i]."</a>";
}
}
if($page != $pages)
{$i=$page+1;
$text2.=<a href='javascript:void(null);' onclick=
    document.forms.FormSearchZakaz.pagesearch.value=". $i .";
    xajax_View_Search_Zakaz_Admin(xajax.getFormValues(
    \"FormSearchZakaz\"));
    '> >></a>";
}
if($pages != 1)
{$text2.=<br><br>Всего - ".$count." Страниц - ".$pages."<br>
    </center>";
}
else
{$text2.=</center>};
}
```

```
    }
}
else
{
    $text2=<br><center>По данному запросу поиска ничего не
        обнаружено</center><br>";
}

$text[0]=$text1;
$text[1]=$text2;
return $text;
}

?>
```

4.4.3. Просмотр заказа

При нажатии на иконку **Подробно** можно просмотреть выбранный заказ из списка (рис. 4.23). При этом вызывается хайп-функция `View_Zakaz_Admin`, расположенная в файле `prgzakaz_admin/view_zakaz_admin.php` (листинг 4.26). В качестве аргумента передается ID заказа в таблице `zakaz` базы данных.

Листинг 4.26

```
<?php
// Просмотр заказа (админ)
// $Id - id заказа
function View_Zakaz_Admin($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // формирование контента для вывода - информация о заказе
    $content=f_view_zakaz_admin($Id);
    // заголовок
    $zagcontent=f_zag1("Просмотр заказа");
    // вывод заголовка
    $objResponse->assign("centercaption5","innerHTML",$zagcontent);
    // вывод контента
    $objResponse->assign("center5","innerHTML",$content);
    $objResponse->script("document.getElementById
        ('center5').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Просмотр заказа				
Заказ 31		Логин 11111	Статус - оплачен	
Товар	Кол-во	Цена, руб	Сумма, руб	
1 База почтовых индексов населенных пунктов России	1	47.50	47.50	
2 Единый федеральный реестр туроператоров	1	22.50	22.50	
3 База предприятий Юг России 2009.	1	50.00	50.00	
Итого	3		120.00	

Рис. 4.23. Просмотр заказа администратором

Контент для вывода данных заказа формирует функция `f_view_zakaz_admin`, расположенная в файле `prgzakaz_admin/function_view_zakaz_admin.php` (листинг 4.27). Позиции заказа функция выбирает из таблицы `zakaz_table`.

Листинг 4.27

```
<?php
// Просмотр заказа (админ)
// $Id - id заказа
function f_view_zakaz_admin($Id)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    $text=array();
    $text1="";
    // выбор заказа
    $query1="SELECT * FROM zakaz WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    $row1=mysql_fetch_assoc($rez1);
    // номер
    $text1.= "<br><b>Заказ</b> ".$row1[id]."<br>";
    $query3="SELECT login FROM users WHERE id='".$row1[id_user]."' ";
    $rez3=mysql_query($query3);
    // логин
    $text1.= "<br><b>Логин</b> ".mysql_result($rez3,0)."<br>";
    // статус оплаты
    if($row1[pay]=='yes')
        $text1.= "<b>Статус </b> - оплачен<br>";
    else
        $text1.= "<b>Статус </b> - не оплачен<br><br>";
```

```
// выбор строк заказа
$query2="SELECT * FROM zakaz_table WHERE id_zakaz='".$Id."'";
$rez2=mysql_query($query2);
// шапка таблицы
$text1."<table>";
$text1."<tr><td class='str0'></td>";
$text1."<td class='str0'>Товар</td>";
$text1."<td class='str0' align=right>Кол-во</td>";
$text1."<td class='str0' align=right>Цена, <br>руб</td>";
$text1."<td class='str0' align=right>Сумма, <br>руб</td>";
$text1."<td class='str0'></td>";
$text1."</tr>";
$i=0;$count=0;
while($row2=mysql_fetch_assoc($rez2))
{
    $i++;
    $text1."<tr><td class='str'.".$i%2+1)." align=right>".$i."</td>";
    $query3="SELECT name FROM tovars WHERE id='".$row2[id_tovar].'";
    // наименование
    $rez3=mysql_query($query3);
    $text1."<td class='str'.".$i%2+1)."!>
        ".mysql_result($rez3,0)."</td>";
    // количество
    $text1."<td class='str'.".$i%2+1)." align=right>
        ".$row2[kol]."</td>";
    // накопление количества
    $count+=$row2[kol];
    // цена товара
    $text1."<td class='str'.".$i%2+1)." align=right>
        ".$row2[pay_rub]."</td>";
    // сумма по позиции
    $text1."<td class='str'.".$i%2+1)." align=right>
        ".$row2[summa_rub]."</td>";
    // ссылка на скачивание
    if($row2[id_link]>0)
        $text1."<td class='str'.".$i%2+1)."!><a href='download.php?
            id=".$row2[id_link]."! target='_blank'>
            <img src='img/download.png'></a></td></tr>";
    else
        $text1."<td class='str'.".$i%2+1)."!></td>";
    $text1."</tr>";
}
$text1."<tr><td></td>";
$text1."<td>Итого</td>";
// вывод итого количества
$text1."<td align=right>".$count."</td>";
```

```

$text1.= "<td align=right></td>";
// вывод итого сумма
$text1.= "<td align=right>".$row1[summa_rub]."</td>";
$text1.= "<td></td></tr>";
if($row1[pay]=='no')
{
    $text1.= "<tr><td></td>";
    $text1.= "<td>Оплачено</td>";
    $text1.= "<td align=right></td>";
    $text1.= "<td align=right></td>";
    $text1.= "<td align=right>".$row1[summa_rub_oplata]."</td>";
    $text1.= "<td></td></tr>";
}
$text1.= "</table>";
// ссылка на установку признака оплаты для неоплаченного заказа
if($row1[pay]=='no')
{
    $text1.= "<br><br><input type=button value='Сделать оплаченным'
        onclick='xajax_Oplata_Zakaz_Admin(\".$row1[id].\") ;' >";
}
return $text1;
}
?>

```

4.4.4. Редактирование заказа

При нажатии на иконку **Редактировать** (для неоплаченных заказов) можно отредактировать выбранный заказ из списка (рис. 4.24). При этом вызывается халф-функция `Edit_Zakaz_Admin`, расположенная в файле `prgzakaz_admin/edit_zakaz_admin.php` (листинг 4.28). В качестве аргумента передается ID заказа в таблице `zakaz` базы данных.

Листинг 4.28

```

<?php
// Редактирование заказа - admin
function Edit_Zakaz_Admin($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // получение контента - формы редактирования заказа
    $content=f_edit_zakaz_admin($Id);
    // заголовок
    $zagcontent=f_zag1("Просмотр заказа");
    // вывод заголовка
    $objResponse->assign("centercaption5","innerHTML",$zagcontent);
    // вывод контента
    $objResponse->assign("center5","innerHTML",$content);
}

```

```

// блок center5 в зону видимости
$objResponse->script("document.getElementById
('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

Товар	Кол-во	Цена, руб	Сумма, руб
1 Справочник калорийности продуктов	1	10.00	10.00
2 База предприятий Юг России 2009.	1	200.00	200.00
3 Телефонные коды городов России	1	10.00	10.00
4 Единый федеральный реестр туроператоров	1	45.00	45.00
5 База предприятий Дальний Восток 2009	1	200.00	200.00
Итого	5		465.00

Рис. 4.24. Редактирование заказа администратором

Контент для вывода данных заказа формирует функция `f_edit_zakaz_admin`, расположенная в файле `prgzakaz_admin/function_edit_zakaz_admin.php` (листинг 4.29). Функция создает форму `FormEditZakaz`, выбирает поля из таблицы `zakaz_table`. Для каждой позиции, для поля **Количество** по событию `onchange` вызываем хайп-функцию `Change_Kol_Table` с передачей данных по данной позиции. Для полей **Цена** по событию `onchange` вызываем хайп-функцию `Change_Pay_Table` с передачей данных по данной позиции. При нажатии иконки **Удалить** вызываем хайп-функцию `Change_Kol_Table` с передачей данных по текущей позиции и количеством, равным нулю.

Листинг 4.29

```

<?php
// Редактирование заказа - admin
// $Id - id заказа
function f_edit_zakaz_admin($Id)

```

```
{  
// подключение файла настроек  
require_once("my.php");  
// подключение базы данных  
require_once("mybaza.php");  
// создание формы  
$text1="";  
$text1.=<form id='FormEditZakaz' action='javascript:void(null);'  
onsubmit='xajax.$(\"ButtonFormEditZakaz\").disabled=true;  
xajax.$(\"ButtonFormEditZakaz\").value=  
\"Подождите...\";  
xajax_Go_Edit_Zakaz_Admin(xajax.getFormValues(  
\"FormEditZakaz\"));'>;  
$text1.=<table width=100%>;  
$query1="SELECT * FROM zakaz WHERE id='".$Id."' ";  
$rez1=mysql_query($query1);  
$row1=mysql_fetch_assoc($rez1);  
$text1.=<br><br><b>Заказ</b> ".$row1[id].<br>;  
$query11="SELECT login FROM users WHERE id='".$row1[id_user']."' ";  
$rez11=mysql_query($query11);  
$text1.=<br><br><b>Клиент - </b> ".mysql_result($rez11,0).<br>;  
if($row1[pay]=='yes')  
    $text1.=<br><b>Статус </b> - оплачен<br>;  
else  
    $text1.=<br><b>Статус </b> - не оплачен<br>;  
$query2="SELECT * FROM zakaz_table WHERE id_zakaz='".$Id."' ";  
$rez2=mysql_query($query2);  
// шапка для таблицы  
$text1.=<table>;  
$text1.=<tr><td class='str0'></td>;  
$text1.=<td class='str0'>Товар</td>;  
$text1.=<td class='str0' align=right>Кол-во</td>;  
$text1.=<td class='str0' align=right>Цена, <br>руб</td>;  
$text1.=<td class='str0' align=right>Сумма, <br>руб</td>;  
$text1.=<td class='str0'></td>;  
$text1.=</tr>;  
$i=0;$count=0;  
// выбор позиций заказа  
while($row2=mysql_fetch_assoc($rez2))  
{  
    {$i++;$count+=$row2[kol];  
    $text1.=<tr>;  
    $text1.=<td class='str'.($i%2+1)."' width=5%>".$i."  
        <input type=hidden name=table_id".$i." id=table_id".$i."  
        value='".$row2[id]."' ></td>;  
    $query1="SELECT name FROM tovars WHERE id='".$row2[id_tovar]."' ";
```

```
$text1.=<td class='str' .($i%2+1)."  
    width=60%>.mysql_result(mysql_query($query1),0)."</td>";  
$text1.=<td class='str' .($i%2+1)."  
    width=10%><input type=text  
    name=table_kol".$row2[id]."  
    id=table_kol".$row2[id]."  
    value='".$row2[kol]."' size=3 maxlength=3 ".READONLYZ."  
    onchange='var x=new Array();x[0]='".$row2[id].";x[1]=this.value;  
    x[2]=document.forms.FormEditZakaz.table_pay".$row2[id].".value;  
    x[3]=document.forms.FormEditZakaz.table_summa".$row2[id].".value;  
    x[4]=document.forms.FormEditZakaz.itogo_count_zakaz.value;  
    x[5]=document.forms.FormEditZakaz.itogo_summa_zakaz.value;  
    xajax_Change_Kol_Table(x);'>  
    </td>";  
  
$text1.=<td class='str' .($i%2+1)."  
    width=10%><input type=text  
    name=table_pay".$row2[id]."  
    id=table_pay".$row2[id]."  
    value='".$row2[pay_rub]."' size=3 maxlength=6  
    onchange='var x=new Array();x[0]='".$row2[id].";  
    x[1]=document.forms.FormEditZakaz.table_kol".$row2[id].".value;  
    x[2]=this.value;  
    x[3]=document.forms.FormEditZakaz.table_summa".$row2[id].".value;  
    x[4]=document.forms.FormEditZakaz.itogo_count_zakaz.value;  
    x[5]=document.forms.FormEditZakaz.itogo_summa_zakaz.value;  
    xajax_Change_Pay_Table(x);'>  
    </td>";  
  
$text1.=<td class='str' .($i%2+1)."  
    width=10%><input type=text  
    name=table_summa".$row2[id]."  
    id=table_summa".$row2[id]."  
    value='".$row2[summa_rub]."' size=6 maxlength=6 readonly  
    onclick='document.getElementById(  
        \'table_kol".$row2[id]."\').focus();return false;'>  
    </td>";  
  
if(READONLYZ=='readonly')  
    $text1.=<td class='str' .($i%2+1)."  
        width=5%><a  
        href='javascript:void();' onclick='  
            var x=new Array();x[0]='".$row2[id].";x[1]='\0\";  
            x[2]=document.forms.FormEditZakaz.table_pay".$row2[id].".value;  
            x[3]=document.forms.FormEditZakaz.table_summa".$row2[id].".value;  
            x[4]=document.forms.FormEditZakaz.itogo_count_zakaz.value;  
            x[5]=document.forms.FormEditZakaz.itogo_summa_zakaz.value;  
            xajax_Change_Kol_Table(x);'><img src='img/delete.png'>  
        </a></td>";  
    $text1.= "</tr>";  
}  
  
$text1.= "<tr><td></td>";  
$text1.= "<td>Итого</td>";  
$text1.= "<td align=right><input type=text name=itogo_count_zakaz  
id=itogo_count_zakaz
```

```

        value='".$count."' size=8 maxlength=8 readonly></td>";
$text1.= "<td align=right></td>";
$text1.= "<td align=right><input type=text name=itogo_summa_zakaz
    id=itogo_summa_zakaz value='".$row1[summa_rub]."' size=8 maxlength=8
    readonly></td>";
$text1.= "<td></td></tr>";
$text1.= "</table>";
$text1.= "<center><input type='submit' id='ButtonFormEditZakaz'
    value='Изменить -'>
<br><br><input type='button' value='Назад' onclick='
    document.getElementById(\"center5\").innerHTML=\"\";
    document.getElementById(\"centercaption5\").innerHTML=\"\";\'>
</center>";
$text1.= "</form>";
return $text1;
}
?>
```

Хајах-функция `Change_Kol_Table` нам уже встречалась (см. листинг 3.71). Рассмотрим функцию `Change_Pay_Table`, которая находится в файле `prgzakaz/change_pay_table.php` (листинг 4.30). Функция получает ID позиции, количество, новую цену, старую сумму позиции и итоговую сумму, пересчитывает и устанавливает новую сумму для позиции и итоговую сумму.

Листинг 4.30

```

<?php
// Изменение цены товара
// в позиции товара при редактировании заказа - admin
function Change_Pay_Table($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // вычислить И установить новое значение в ячейку суммы
    $new_summa=$Id[1]*$Id[2];
    $objResponse->assign("table_summa".$Id[0],"value",$new_summa);
    // вычислить изменение количества и суммы
    $new_itogo_summa=$Id[5]-$Id[3]+$Id[1]*$Id[2];
    $objResponse->assign("itogo_summa_zakaz","value",$new_itogo_summa);
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

При нажатии на кнопку **Изменить** из файла `prgzakaz_admin/go_edit_zakaz_admin.php`, (листинг 4.31) происходит вызов хајах-функции `Go_Edit_Zakaz`

с передачей ей параметров формы FormEditZakaz. Функция перезаписывает в базу данных значения количества и цены для каждой позиции заказа в таблице zakaz_table, а также изменившуюся сумму заказа в таблице zakaz. При успешном изменении заказа выдается сообщение **Изменено** (рис. 4.25).

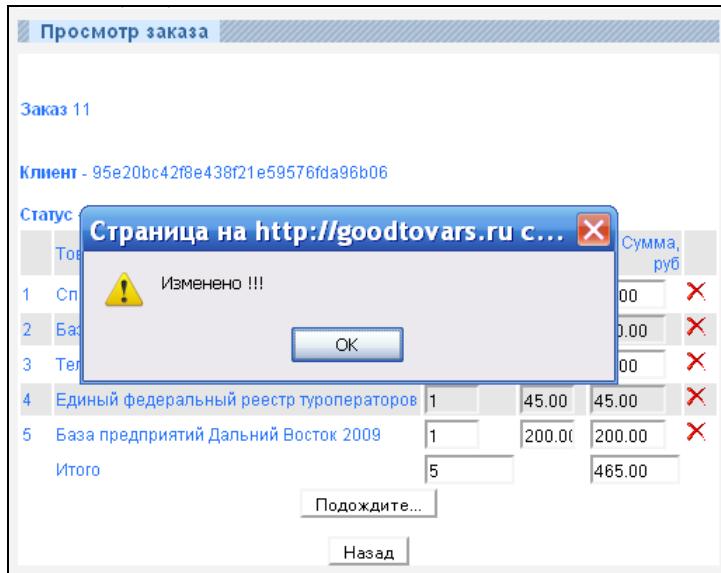


Рис. 4.25. Сообщение при сохранении результатов редактирования

Листинг 4.31

```
<?php
// Сохранение отредактированного заказа
function Go>Edit_Zakaz($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    $j=(count($Id)-2)/4;
    for ($i=1;$i<=$j;$i++)
    {
        $id=$Id[table_id.$i];
        $kol=$Id[table_kol.$id];
        $summa_rub=$Id[table_summa.$id];
        $query1="UPDATE zakaz_table SET kol='".$kol."',
            summa_rub='".$summa_rub."'
            WHERE id='".$id."' ";
```

```

$rez1=mysql_query($query1);
}
$objResponse->alert("Изменено !!!");
// изменение суммы заказа в таблице zakaz
$query2="SELECT id_zakaz FROM zakaz_table WHERE id='".$id."'";
$id_zakaz=mysql_result(mysql_query($query2),0);
$query3="UPDATE zakaz SET summa_rub='".$Id[itogo_summa_zakaz]."'"
WHERE id='".$id_zakaz."'";
mysql_query($query3);
// вывести измененный заказ
$content=f_edit_zakaz($id_zakaz);
$zagcontent=f_zag1("Просмотр заказа");
$objResponse->assign("centercaption5","innerHTML",$zagcontent);
$objResponse->assign("center5","innerHTML",$content);
$objResponse->script("document.getElementById
('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

4.4.5. Удаление заказа

При нажатии на иконку **Удалить** выбранный заказ удаляется. При этом вызывается хайп-функция `Delete_Zakaz_Admin`, расположенная в файле `prgzakaz_admin/delete_zakaz_admin.php` (листинг 4.32). В качестве аргумента передается ID заказа в таблице `zakaz` базы данных. При успешном удалении выдается сообщение **Заказ удален** (рис. 4.26). Из базы данных заказ не удаляется, для него в таблице `zakaz` устанавливается значение поля `visible=no`, и он становится невидимым при просмотре.

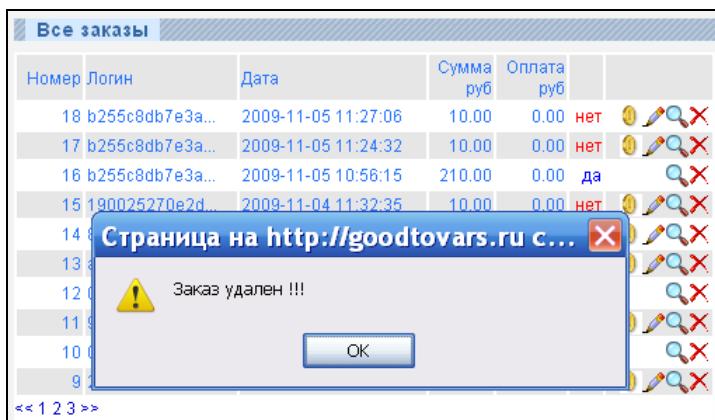


Рис. 4.26. Сообщение при удалении заказа

Листинг 4.32

```
<?php
// Удаление заказа
function Delete_Zakaz ($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // установка поля visible=no
    $query1="UPDATE zakaz SET visible='no' WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    if($rez1)
    {
        $objResponse->alert ("Заказ удален !!!");
        // вывести список заказов
        $content=f_view_all_zakaz(1);
        $objResponse->assign("center3","innerHTML",$content[0]);
        $objResponse->assign("center4","innerHTML",$content[1]);
        $objResponse->script ("document.getElementById
            ('center3').scrollIntoView();");
        $objResponse->assign("centercaption5","innerHTML","<table></table>");
        $objResponse->assign("center5","innerHTML","<table></table>");
    }
    else
        $objResponse->alert ("Ошибка удаления !!!");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

4.4.6. Оплата заказа администратором

Создадим возможность для администратора переводить заказ в статус "оплачен". Для этого достаточно установить для заказа в базе данных в таблице *zakaz* значение поля *pay=yes*. При нажатии на иконку **Оплатить** идет вызов хайло-функции *Oplata_Zakaz_Admin*, которая находится в файле *prgoplata/oplata_zakaz_admin.php* (листинг 4.33). Функция устанавливает значение поля *pay=yes* и выдает сообщение об успешной операции (рис. 4.27). Не забываем также сформировать ссылки на скачивание для каждой позиции заказа в таблице *link_downloads* и вызвать функцию *create_htaccess* для формирования доступа к файлам скачивания для пользователя, создавшего заказ.

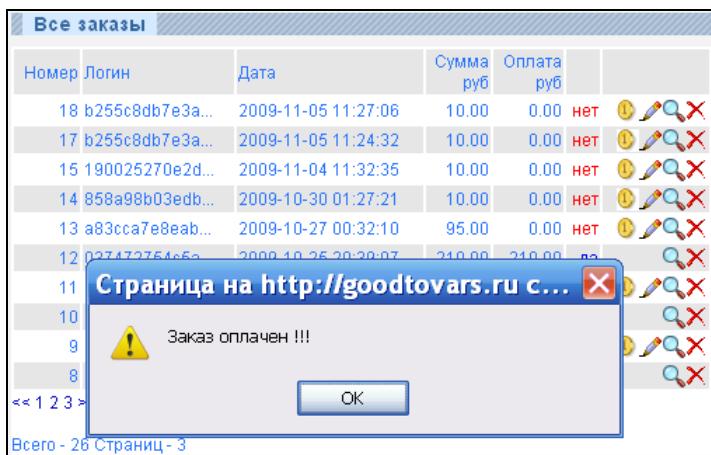


Рис. 4.27. Сообщение при оплате заказа администратором

Листинг 4.33

```
<?php
// Сделать заказ оплаченным admin
// pay=yes
function Oplata_Zakaz_Admin($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // установка признака оплаты
    $query1="UPDATE zakaz SET pay='yes' WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    // создание ссылок на скачивание
    $data=date('Y-m-d',strtotime('now+10days'));
    $query2="SELECT id,id_tovar FROM zakaz_table WHERE id_zakaz='".$Id."'";
    $rez2=mysql_query($query2);
    while($row2=mysql_fetch_assoc($rez2))
    {
        $query3="SELECT arhiv FROM tovars WHERE id='".$row2[id_tovar]."' ";
        $link=mysql_result(mysql_query($query3),0);
        $query4="SELECT id_user FROM zakaz WHERE id='".$Id."'";
        $id_user=mysql_result(mysql_query($query4),0);
        $query5="INSERT INTO link_downloads SET id_user='".$id_user."',
        file='".$link."',status='yes',data='".$data."',id_zakaz='".$Id."'";
        $rez5=mysql_query($query5);
        $id_link=mysql_insert_id();
    }
}
```

```
$query6="UPDATE zakaz_table SET id_link='".$id_link."'";
      WHERE id='".$row2[id]."' ";
$rez6=mysql_query($query6);
}

// изменения в файле .htaccess
create_new_htaccess();
// вывод результата
if($rez1)
{
    $objResponse->alert("Заказ оплачен !!!");
    // заново вывести заказы
    $content.=f_view_zakaz_admin($Id);
    $objResponse->assign("center5","innerHTML",$content);
    $objResponse->script("document.getElementById
        ('center5').scrollIntoView();");
}
else
{
    $objResponse->alert("Ошибка оплаты admin !!!");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

4.5. Операции с профилями пользователей

Управление пользователями включает следующий функционал:

- просмотр пользователей;
- поиск пользователей;
- изменение данных;
- просмотр операций пользователей.

4.5.1. Просмотр всех пользователей

При нажатии ссылки **Пользователи** в главном меню, получаем список всех пользователей постранично (рис. 4.28). Для каждого пользователя будем выводить следующие данные:

- дату регистрации;
- логин;
- тип пользователя:
 - админ — администратор;
 - пользователь — зарегистрированный пользователь;
 - гость — незарегистрированный пользователь;

статус:

- yes — активирован;
- no — неактивирован;
- del — удален.

список IP-адресов;

ссылки:

- подробно;
- редактировать;
- блокировать.

Все пользователи						
Id	Дата	Логин	руб	Тип	Статус	ip
1	0000-00-00 00:00:00	11111		User	yes	195.186.219.71 177.39.66.172 195.186.219.132 194.186.219.71
2	0000-00-00 00:00:00	99999		Админ	yes	127.0.0.1 77.39.66.172 194.186.219.71 194.186.219.132 109.168.135.236
19	0000-00-00 00:00:00	666666		User	yes	
35	0000-00-00 00:00:00	888888		User	yes	127.0.0.1
37	2009-10-14 11:49:17	280a865fce5c...		Гость	yes	127.0.0.1
38	2009-10-14 11:56:54	777777		User	yes	127.0.0.1 194.186.219.78
39	2009-10-15 15:50:14	76d52a30451c...		Гость	yes	127.0.0.1
40	2009-10-18 00:58:42	f9016712e67f...		Гость	yes	77.39.66.172
41	2009-10-19 09:02:23	c4aba755669a...		Гость	yes	194.186.219.71
42	2009-10-19 09:11:05	15ccb334efa...		Гость	yes	194.186.219.71
1	2	3	4	5	6	7 ... 169 170 >>
Всего - 1699 Страниц - 170						

Рис. 4.28. Список всех пользователей

Функция `f_view_all_users` формирует контент для вывода всех пользователей постранично, получая в качестве аргумента номер страницы для вывода. Данные берутся из таблицы `users`. Функция расположена в файле `prgusers_admin/function_view_all_users.php` (листинг 4.34).

Листинг 4.34

```
<?php
// Просмотр пользователей постранично (админ)
// $Id - номер страницы для показа
function f_view_all_users($Id)
```

```
{  
require_once("my.php");  
require_once("mybaza.php");  
$text=array();  
$text1="";  
// подсчет количества всех пользователей  
$query0="SELECT COUNT(id) FROM users ORDER BY id ASC ";  
$rez0=mysql_query($query0);  
$count=mysql_result($rez0,0);  
$pages=ceil($count/NN4);  
$page=min($Id,$pages); $poz=($page-1)*NN4;  
$text1.=<div class='zag_view_tovars'>;  
if($count>0)  
{  
// выбор для нужной страницы  
$query1="SELECT * FROM users ORDER BY id  
ASC LIMIT ".$poz.", ".NN4."";  
$rez1=mysql_query($query1);  
// вывод шапки для таблицы вывода  
$text1.=<table>;  
$text1.=<tr><td class='str0' align=right>Id</td>;  
$text1.=<td class='str0'>Дата</td>;  
$text1.=<td class='str0' align=right>Логин<br>руб</td>;  
$text1.=<td class='str0' align=right>Тип</td>;  
$text1.=<td class='str0' align=right>Статус</td>;  
$text1.=<td class='str0' align=right>ip</td>;  
$text1.=<td class='str0'></td></tr>; $i=0;  
$arrtype=array("", "Гость", "User", "", "", "", "", "", "", "Админ");  
while($row1=mysql_fetch_assoc($rez1))  
{ $i++;  
$text1.=<tr><td class='str'.($i%2+1)."' align=right>  
".$row1[id]."</td>;  
// дата регистрации  
$text1.=<td class='str'.($i%2+1)."'>".$row1[data]."</td>;  
$login=$row1[login];  
$login=(strlen($login)<13)?($login):(substr($login,0,12)."....");  
// логин  
$text1.=<td class='str'.($i%2+1)."' align=right>".$login."</td>;  
// тип пользователя  
$text1.=<td class='str'.($i%2+1)."' align=right>  
".$arrtype[$row1[type]]."</td>;  
// статус  
$text1.=<td class='str'.($i%2+1)."' align=right>  
".$row1[visible]."</td>;  
// ip адреса
```

```
$text1.= "<td class='str' .($i%2+1).'" align=right>
    ".str_replace(";", "<br>", $row1[ip])."</td>";
$text1.= "<td class='str' .($i%2+1).'" align=right>";
// ссылки
$text1.= "<a href='javascript:void();' onclick='
    xajax_View_User_Admin(\".$row1[id]\");'
    title='Подробно'><img src='img/view.gif'></a>";
$text1.= "<a href='javascript:void();' onclick='
    xajax_Edit_User_Admin(\".$row1[id]\");'
    title='Редактировать'><img src='img/edit.png'></a>";
$text1.= "<a href='javascript:void();' onclick='
    xajax_Delete_User_Admin(\".$row1[id]\");'
    title='Блокировать'><img src='img/delete.png'></a>";
$text1.= "</td></tr>";
}
$text1.= "</table>";
// список ссылок перехода по страницам
$text2="";
if ($pages>1)
{
if ($page != 1)
{$i=$page-1;
$text2.= "<a href='javascript:void(null);' onclick='
    var x=new Array();x=\"." .($page-1). ";
    xajax_View_All_Users(x);'> <<</a>";
}
$x=array();
$x=doarray1($page,$pages,5);
for ($i=0;$i < count($x);$i++)
{
if ($x[$i]==$page)
    $text2.= "<a> ".$x[$i]."</a>";
else
{
$text2.= "<a href='javascript:void(null);' onclick='
    var x=new Array();x=\"$x[$i].";
    xajax_View_All_Users(x);'> ".$x[$i]."</a>";
}
}
if ($page != $pages)
{$i=$page+1;
$text2.= "<a href='javascript:void(null);' onclick='
    var x=new Array();x=\"." .($page+1). ";
    xajax_View_All_Users(x);'> >></a>";
}
}
```

```
if($pages != 1)
{$text2.= "<br><br>Всего - ".$count." Страниц - ".$pages."<br>
</center>";}
else
{$text2.= "</center>";}
}
}
else
{
$text1=<br><center>Пользователей не обнаружено</center><br>";
}
$text[0]=$text1;
$text[1]=$text2;
return $text;
}
?>
```

4.5.2. Просмотр пользователей по фильтру

Форма поиска пользователей по фильтру приведена на рис. 4.29. Ее контент формирует функция `f_form_search_users`, расположенная в файле `prgusers_admin/function_form_search_users.php` (листинг 4.35). Форма позволяет осуществлять поиск пользователей по:

- логину;
- IP-адресу;
- типу пользователя;
- статусу;
- поиск пользователя, купившего товар (поиск по наименованию товара).

Листинг 4.35

```
<?php
// Форма поиска пользователей
function f_form_search_users()
{
// подключение файла настроек
require_once("my.php");
// подключение к базе данных
require_once("mybaza.php");
$text1.= "<form id='FormSearchUsers' action='javascript:void(null);'
onsubmit='xajax.\\"ButtonFormSearchUsers\\").disabled=true;
xajax.\\"ButtonFormSearchUsers\\").value=\"Подождите...\";
xajax_View_Search_Users(xajax.getFormValues(
```

```
    \\"FormSearchUsers\\");'>;

$text1.=<table width=100%>;
$text1.=<tr><td width=50%><b>ФИЛЬТР ПО</b></td></tr>";
// логин
$text1.=<tr><td width=50%>Логин</td>";
$text1.=<td width=50%
    <input type='text' name='login' value='' size=32 maxlength=32>
    <input type='hidden' id='pagesearch' name='pagesearch'
    value='1'></td></tr>";
$text1.=<tr><td width=50%>IP - адрес</td>";
// ip адрес
$text1.=<td width=50%
    <input type='text' name='ip' value='' size=16
    maxlength=16></td></tr>";
// тип пользователя
$text1.=<tr><td width=50%>Тип</td>";
$text1.=<td width=50%><select name=type>
    <option value='0' selected>Бсе
    <option value='1'>Гость
    <option value='2'>User
    <option value='9'>Админ
    </select></td></tr>";
// статус
$text1.=<tr><td width=50%>Статус</td>";
$text1.=<td width=50%><select name=visible>
    <option value='all' selected>Бсе
    <option value='yes'>yes
    <option value='no'>no
    <option value='block'>block
    </select></td></tr>";
// купившие товар
$text1.=<tr><td width=50%><b>ИЛИ </b></td></tr>";
$text1.=<tr><td width=50%>Купившие товар ( наименование)</td>";
$text1.=<td width=50%
    <input type='text' name='name_tovar' value='' size=20
    maxlength=20></td></tr>";
$text1.=("</table>";
$text1.= "<center><br><input type='submit' id='ButtonFormSearchUsers'
    value='Найти ->'></center>";
$text1.= "</form>";
return $text1;
}

?>
```

Форма поиска пользователей по фильтру. Виджеты включают: поле ввода для фильтра по имени пользователя; поле ввода для IP-адреса; выпадающее меню для выбора типа (все или база); выпадающее меню для выбора статуса (да или нет); поле ввода для наименования товара; и кнопку поиска "Найти ->".

Рис. 4.29. Форма поиска пользователей по фильтру

При нажатии на кнопку **Найти** из файла prgusers_admin/view_search_users.php, (листинг 4.36) вызывается хайп-функция View_Search_Users, которой передаются значения формы FormSearchUsers.

Листинг 4.36

```
<?php
// Просмотр пользователей по поиску
function View_Search_Users ($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // заголовок
    $zag=f_zag1("Результат поиска пользователей");
    // вывод заголовка
    $objResponse->assign("centercaption3","innerHTML",$zag);
    // формирование контента – результатов поиска
    $content=f_view_search_users($Id);
    // вывод контента
    $objResponse->assign("center3","innerHTML",$content[0]);
    // вывод навигатора страниц
    $objResponse->assign("center4","innerHTML",$content[1]);
    // блок center3 в зону видимости
    $objResponse->script ("document.getElementById
        ('center3').scrollIntoView();");
    // активировать кнопку
    $objResponse->assign("ButtonFormSearchUsers","value", "Найти ->");
    $objResponse->assign("ButtonFormSearchUsers","disabled", false);
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Результаты поиска по фильтру формирует функция `f_view_search_users`, расположенная в файле `prgusers_admin/function_view_search_users.php` (листинг 4.37).

Листинг 4.37

```
<?php
// Просмотр пользователей по фильтру
// $Id - параметры формы
function f_view_search_users($Id)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
    $text=array();
    $text1="";
    $name_tovar=utftowin($Id[name_tovar]);
    $login=utftowin($Id[login]);
    $ip=utftowin($Id[ip]);
    $type=$Id[type];
    $visible=$Id[visible];
    // создание запроса
    if(strlen(rtrim(ltrim($name_tovar)))>0)
        { // по наименованию товара
        $query0="SELECT DISTINCT(users.id) FROM
            zakaz,zakaz_table,tovars,users
            WHERE zakaz.visible='yes' && zakaz.id_user=users.id &&
            LOWER(tovars.name) LIKE '%".$name_tovar."%' &&
            zakaz_table.id_tovar=tovars.id && zakaz_table.id_zakaz=zakaz.id ";
        }
    else
        { // другие параметры
        $query0="SELECT id FROM users WHERE id>0 ";
        if(strlen(rtrim(ltrim($login)))>0)
            {
            $query0.="&& LOWER(login) LIKE '%".$login."%' ";
            }
        if(strlen(rtrim(ltrim($ip)))>0)
            {
            $query0.="&& ip LIKE '%".$ip."%' ";
            }
        if($type>0)
            {
            $query0.="&& type='".$type."'";
            }
        }
}
```

```
if($visible!='all')
{
$query0="&& visible='".$visible."' ";
}
}

$rez0=mysql_query($query0);
$count=mysql_num_rows($rez0);
$pages=ceil($count/NN4);
$page=min($Id[pagesearch],$pages);$poz=($page-1)*NN4;
$text1=<div class='zag_view_tovars'>";
if($count>0)
{ // непустой результат
$query0=$query0." LIMIT ".$poz.", ".NN4." ";
$rez0=mysql_query($query0);
// шапка таблицы результата поиска
$text1.= "<table>";
$text1.= "<tr><td class='str0' align=right>Id</td>";
$text1.= "<td class='str0'>Дата</td>";
$text1.= "<td class='str0' align=right>Логин<br>руб</td>";
$text1.= "<td class='str0' align=right>Тип</td>";
$text1.= "<td class='str0' align=right>Статус</td>";
$text1.= "<td class='str0' align=right>ip</td>";
$text1.= "<td class='str0'></td></tr>";$i=0;
$arrtype=array("", "Гость", "User", "", "", "", "", "", "Админ");
while($row0=mysql_fetch_row($rez0))
{$i++;
$query1="SELECT * FROM users WHERE id='".$row0[0]."' ";
$rez1=mysql_query($query1);
$row1=mysql_fetch_assoc($rez1);
$text1.= "<tr><td class='str".($i%2+1)."'"
align=right>".$row1[id]."</td>";
// дата регистрации
$text1.= "<td class='str".($i%2+1)."'>".$row1[data]."</td>";
$login=$row1[login];
$login=(strlen($login)<13)?($login):(substr($login,0,12)."...");
// логин
$text1.= "<td class='str".($i%2+1)."' align=right>".$login."</td>";
// тип пользователя
$text1.= "<td class='str".($i%2+1)."'"
align=right>".$arrtype[$row1[type]]."</td>";
// статус пользователя
$text1.= "<td class='str".($i%2+1)."'"
align=right>".$row1[visible]."</td>";
// ip адреса пользователя
$text1.= "<td class='str".($i%2+1)."'"
```

```
        align=right>".str_replace(";", "<br>", $row1[ip])."</td>";
$text1.= "<td class='str' .($i%2+1).'" align=right>";
// ссылки
$text1.= "<a href='javascript:void();' onclick='
    xajax_View_User_Admin(\".$row1[id]\");
    title='Подробно'><img src='img/view.gif'></a>";
$text1.= "<a href='javascript:void();' onclick='
    xajax_Edit_User_Admin(\".$row1[id]\");
    title='Редактировать'><img src='img/edit.png'></a>";
$text1.= "<a href='javascript:void();' onclick='
    xajax_Delete_User_Admin(\".$row1[id]\");
    title='Блокировать'><img src='img/delete.png'></a>";
$text1.= "</td></tr>";
}
$text1.= "</table>";
// список ссылок перехода по страницам
$text2="";
if($pages>1)
{
if($page != 1)
{$i=$page-1;
$text2.= "<a href='javascript:void(null);' onclick='
    document.forms.FormSearchUsers.pagesearch.value=\"$i.\";
    xajax_View_Search_Users(xajax.getFormValues(
        \"FormSearchUsers\"));
    '> <<</a>";
}
$x=array();
$x=doarray1($page,$pages,5);
for($i=0;$i < count($x);$i++)
//for($i=1;$i <= $pages;$i++)
{
if($x[$i]==$page)
$text2.= "<a> ".$x[$i]."</a>";
else
{
$text2.= "<a href='javascript:void(null);' onclick='
    document.forms.FormSearchUsers.pagesearch.value=\"$x[$i]\";
    xajax_View_Search_Users(xajax.getFormValues(
        \"FormSearchUsers\"));
    '> ".$x[$i]."</a>";
}
}
if($page != $pages)
{$i=$page+1;
$text2.= "<a href='javascript:void(null);' onclick='
    document.forms.FormSearchUsers.pagesearch.value=\"$i.\";
    xajax_View_Search_Users(xajax.getFormValues(
        \"FormSearchUsers\"));
    '> ".$i."</a>";
}
}
```

```
    xajax_View_Search_Users(xajax.getFormValues(
        \"FormSearchUsers\"));'> >></a>";
    }
    if($pages != 1)
    {$text2.= "<br><br>Всего - ".$count." Страниц - ".$pages."<br>
        </center>";}
    else
    {$text2.="</center>"}
}
}
else
{
$text2="<br><center>По данному запросу поиска ничего не
обнаружено</center><br>";
}
$text[0]=$text1;
$text[1]=$text2;
return $text;
}
?>
```

4.5.3. Просмотр профиля пользователя

При щелчке по иконке **Подробно** можно просмотреть данные профиля выбранного пользователя (рис. 4.30). При этом вызывается хайп-функция `View_User_Admin`, расположенная в файле `prgusers_admin/view_user_admin.php` (листинг 4.38). В качестве аргумента функция получает ID пользователя в таблице `users`.

Листинг 4.38

```
<?php
// Просмотр данных пользователя
// $Id - id пользователя
function View_User_Admin($Id)
{
$objResponse = new xajaxResponse();
$objResponse->assign("flag_ajax","value",'yes');
// получить контент - данные пользователя
$content=f_view_user_admin($Id);
// заголовок
$zagcontent=f_zag1("Просмотр данных пользователя");
// выдать заголовок
$objResponse->assign("centercaption5","innerHTML",$zagcontent);
// выдать контент
$objResponse->assign("center5","innerHTML",$content);
// блок center5 в зону видимости
```

```

$objResponse->script("document.getElementById
    ('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

Просмотр данных пользователя	
Логин	11111
Пароль	11111
Дата регистрации	0000-00-00 00:00:00
Тип пользователя	User
Статус	yes
E-mail	
Визитов	31
IP-адреса	195.186.219.71 177.39.66.172 195.186.219.132 194.186.219.71
Скидка	50 %
Заказов, всего	4
Заказов оплаченных	3

Рис. 4.30. Просмотр данных профиля пользователя

Контент формирует функция `f_view_user_admin`, расположенная в файле `prgusers_admin/function_view_user_admin.php` (листинг 4.39). Данные берутся из таблиц `users` и `zakaz`. Для каждого пользователя выводится следующая информация:

- логин;
- пароль;
- дата регистрации;
- тип пользователя;
- статус пользователя;
- e-mail пользователя;
- количество визитов;
- IP-адреса пользователя;
- скидка на товары;
- число сделанных заказов;
- число оплаченных заказов.

Листинг 4.39

```

<?php
// Просмотр данных пользователя
// $Id - id пользователя
function f_view_user_admin($Id)

```

```
{  
require_once("my.php");  
require_once("mybaza.php");  
$text=array();  
$text1="";  
// получение данных из таблицы users  
$query1="SELECT * FROM users WHERE id='".$Id."' ";  
$rez1=mysql_query($query1);  
$row1=mysql_fetch_assoc($rez1);  
$text1.="

|                    |                                          |
|--------------------|------------------------------------------|
| Логин              | .\$row1[login].                          |
| Пароль             | .\$row1[password].                       |
| Дата регистрации   | .\$row1[data].                           |
| Тип пользователя   | .\$arrtype[\$row1[type]].                |
| Статус             | .\$row1[visible].                        |
| E-mail             | .\$row1[email].                          |
| Визитов            | .\$row1[vizits].                         |
| IP-адреса          | .\$str_replace(';', '<br>', \$row1[ip]). |
| Скидка             | .\$row1[discount]. %                     |
| Заказов, всего     | .".mysql_result(\$rez2,0)."              |
| Заказов оплаченных | .".mysql_result(\$rez3,0)."              |

";  
return $text1;  
}  
?  
?
```

4.5.4. Редактирование профиля пользователя

При щелчке по иконке **Подробно** откроется форма просмотра и редактирования данных профиля выбранного пользователя (рис. 4.31). При этом вызывается хайло-функция `Edit_User_Admin`, расположенная в файле `prgusers_admin/edit_user_admin.php` (листинг 4.40). В качестве аргумента функция получает ID пользователя в таблице `users`.

Листинг 4.40

```
<?php
// Редактирование данных пользователя
// $Id - id пользователя
function Edit_User_Admin($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // получить контент - форма данных пользователя
    $content=f_edit_user_admin($Id);
    // заголовок
    $zagcontent=f_zag1("Редактирование данных пользователя");
    // вывод заголовка
    $objResponse->assign("centercaption5","innerHTML",$zagcontent);
    // вывод контента
    $objResponse->assign("center5","innerHTML",$content);
    // блок center5 в зону видимости
    $objResponse->script("document.getElementById
        ('center5').scrollIntoView();");
    $objResponse->assign("flag_ajax","value",'no');
    return $objResponse;
}
?>
```

Контент формирует функция `f_view_edit_admin`, расположенная в файле `prgusers_admin/function_edit_user_admin.php` (листинг 4.41). Данные берутся из таблиц `users` и `zakaz`. Список редактируемых полей:

- пароль;
- статус пользователя;
- e-mail пользователя;
- скидка на товары.

Листинг 4.41

```
<?php
// Редактирование данных пользователя
// $Id - id пользователя
function f_edit_user_admin($Id)
{
    // подключение файла настроек
    require_once("my.php");
    // подключение к базе данных
    require_once("mybaza.php");
```

```
$text=array();
$text1="";
// получение данных пользователя
$query1="SELECT * FROM users WHERE id='".$Id."'";
$rez1=mysql_query($query1);
$row1=mysql_fetch_assoc($rez1);
$text1.="";
$text1.="|  |  |
| --- | --- |
| Дата регистрации | .$row1[data]. |
";
$arrtype=array("", "Гость", "User", "", "", "", "", "", "Админ");
$text1.="| Тип пользователя | .$arrtype[$row1[type]]. |
";
// статус
$text1.="| Статус | "; $text1.=" |
";
$text1.="| Визитов | .$row1[vizits]. |
";
$text1.="| Скидка | IP-адреса     </td><td>".str_replace(';', ',<br>', $row1[ip])." |
";
$text1.="| Заказов, всего | </td><td>".mysql_result($rez2, 0)." |
";
$query3="SELECT COUNT(id) FROM zakaz WHERE id_user='".$Id."'";
$rez2=mysql_query($query3);
$text1.="| Заказов | .".mysql_result($rez3, 0)." |
";
```

```

    pay='yes' ";
$rez3=mysql_query($query3);
$text1.= "<tr><td>Заказов оплаченных
</td><td>".mysql_result($rez3,0)."</td></tr>";
$text1.= "</table>";
$text1.= "<center><input type='submit' id='ButtonFormEditUser'
value='Изменить ->'>
<br><br><input type='button' value='Назад' onclick='
document.getElementById(\"center5\").innerHTML=\"\";
document.getElementById(\"centercaption5\").innerHTML=
\"\"; '></center>";
return $text1;
}
?>

```

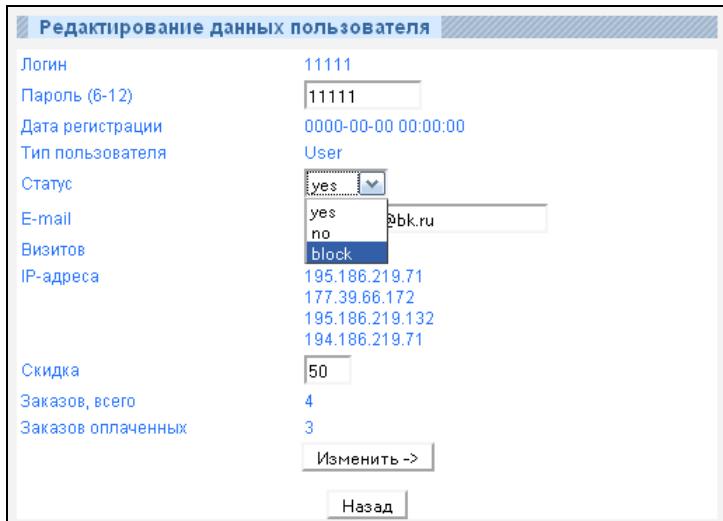


Рис. 4.31. Просмотр данных профиля пользователя

При нажатии кнопки **Изменить** вызывается хайп-функция Go>Edit>User\Admin, расположенная в файле prgusers_admin/go_edit_user_admin.php (листинг 4.42). В функцию передаются данные формы FormEditUser и сохраняются в базе данных. При успешном изменении данных выводится соответствующее сообщение (рис. 4.32) и выдаются новые данные пользователя для просмотра.

Листинг 4.42

```

<?php
// Сохранение отредактированных
// данных пользователя
function Go>Edit>User\Admin($Id)
{

```

```
$objResponse = new xajaxResponse();
$objResponse->assign("flag_ajax","value",'yes');
// подключение базы данных
require_once("mybaza.php");
// изменение данных
$query1="UPDATE users SET password='".$Id[password]."',
    email='".$Id[email]."',discount='".$Id[discount]."',
    visible='".$Id[visible]."' WHERE id='".$Id[id]."' ";
if(mysql_query($query1))
{
    $objResponse->alert("Данные изменены !!!");
    $content=f_view_user_admin($Id[id]);
    $zagcontent=f_zag1("Просмотр данных пользователя");
}
else
{
    $objResponse->alert("Ошибка изменения данных пользователя!!!!");
    $content=f_edit_user_admin($Id[id]);
    $zagcontent=f_zag1("Редактирование данных пользователя");
}
$objResponse->assign("centercaption5","innerHTML",$zagcontent);
$objResponse->assign("center5","innerHTML",$content);
$objResponse->script("document.getElementById
    ('center5').scrollIntoView();");
$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
```

?>

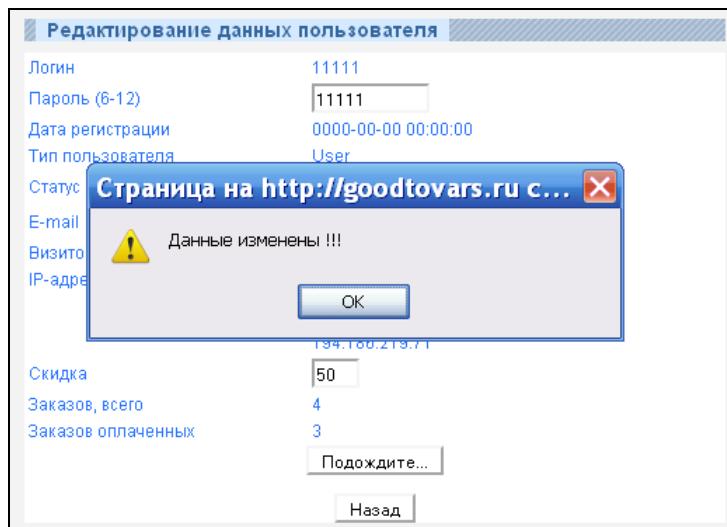


Рис. 4.32. Успешное изменение данных профиля пользователя

4.5.5. Блокировка пользователя

Блокировать пользователя можно при редактировании его профиля, а можно и по ссылке **Блокировать** в списке пользователей. При этом вызывается хайлоуд-функция `Delete_User_Admin`, расположенная в файле `prgusers_admin/delete_user_admin.php` (листинг 4.43). В качестве аргумента передается ID пользователя в таблице `users`. Функция изменяет содержимое поля `visible=block`. При успешном изменении выводится сообщение об этом (рис. 4.33).

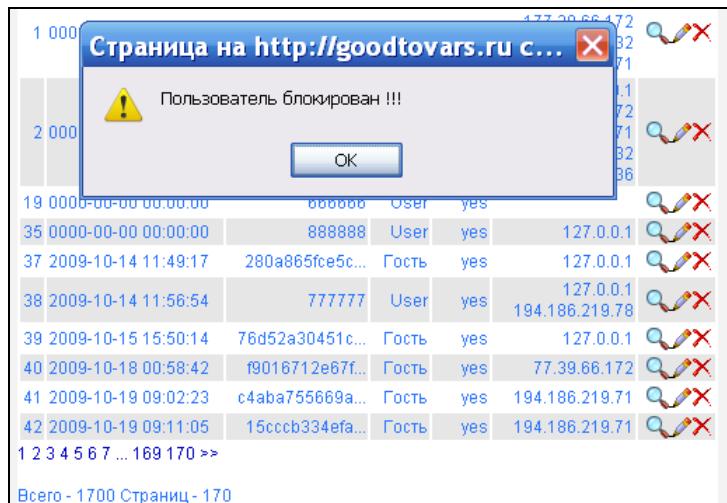


Рис. 4.33. Успешная блокировка пользователя

Листинг 4.43

```
<?php
// Блокировка пользователя
// visible=block
function Delete_User_Admin($Id)
{
    $objResponse = new xajaxResponse();
    $objResponse->assign("flag_ajax","value",'yes');
    // подключение к базе данных
    require_once("mybaza.php");
    // изменение статуса пользователя в таблице users
    $query1="UPDATE users SET visible='block' WHERE id='".$Id."'";
    $rez1=mysql_query($query1);
    if($rez1)
        $objResponse->alert("Пользователь блокирован !!!");
    else
        $objResponse->alert("Ошибка блокировки пользователя!!!");
```

```

$objResponse->assign("flag_ajax","value",'no');
return $objResponse;
}
?>

```

4.6. Обратная связь

Ссылки для обратной связи по e-mail и ICQ расположены в блоке **Контакты** (рис. 4.34).

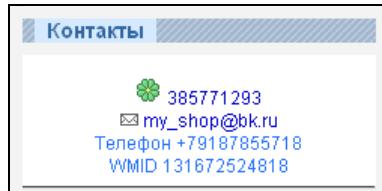


Рис. 4.34. Блок Контакты

4.6.1. Обратная связь по e-mail

При нажатии на ссылку **my_shop@bk.ru** вызывается хайп-функция **Form_Email_Admin**, расположенная в файле **prgcontacts/form_email_admin.php** (листинг 4.44). Функция делает видимым окно (**div id=windowdop**) и создает в нем форму отправки сообщения на e-mail администратора (рис. 4.35).

ЗАМЕЧАНИЕ

Названия ссылок в блоке **Контакты** берутся из файла **my.php**: для e-mail — константа **EMAILADMIN**, для ICQ — константа **ICQADMIN**.

Рис. 4.35. Форма обратной связи — отправка на e-mail администратора

Листинг 4.44

```
// Форма посылки письма админу с сайта
function Form_Email_Admin()
{
    $objResponse = new xajaxResponse();
    require_once("my.php");
    require_once("mybaza.php");
    // сделать окно невидимым
    $objResponse->assign("windowdop","style.display","none");
    // ссылка на закрытие окна
    $text1.=<a href='javascript:void(null);' onclick='
        document.getElementById(\"windowdop\").style.display=
        \"none\";return false;'>
        <img src='img/delete.png' align=right></a>;
    $text1.=<center><b>Отправить вопрос<br>по e-mail</b><br>;
    $text1.=<div id=VhodError></div>;
    $text1.=<form id='FormEmailAdmin' action='javascript:void(null);'
        onsubmit='xajax.$(\"ButtonFormEmailAdmin\").disabled=true;
        xajax.$(\"ButtonFormEmailAdmin\").value=\"Подождите...\";
        xajax_Go_Email_Admin(xajax.getFormValues(
        \"FormEmailAdmin\"));'>;
    $text1.=<font color=black> Кому : Администратору сайта </font><input
        type='hidden' name='toemail' value='".$EMAILADMIN."'><br>;
    $text1.=<font color=black>От(e-mail) : </font><input type='text'
        name='fromemail' value='''><br>;
    $text1.=<font color=black>Тема : <br></font><input type='text'
        name='fromemailtheme' value='Вопрос' size=40 maxlength=40><br>;
    $text1.=<font color=black>Сообщение:<br></font><textarea
        name='fromemailbody' cols=40 rows=7></textarea><br>;
    $text1.=<input id='ButtonFormEmailAdmin' type='submit'
        value='Отправить>>'>;
    $text1.=</center></form>;
    // сделать окно видимым
    $objResponse->script("document.getElementById
        ('windowdop').style.display='block'");
    $objResponse->assign("windowdop","innerHTML",$text1);
    $objResponse->script("document.getElementById
        ('windowdop').scrollIntoView();");
    return $objResponse;
}
// отправка письма
function Go_Email_Admin($Id)
{
    $objResponse = new xajaxResponse();
```

```
if(!ereg("^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.( [a-z]
{2,4})$",trim($Id[fromemail])))
{
$objResponse->assign("ButtonFormEmailAdmin","disabled",false);
$objResponse->assign("ButtonFormEmailAdmin","value",
                      "Отправить>>");
$objResponse->alert("Неправильный e-mail отправки !!!");
return $objResponse;
}

if(trim($Id[frommailtheme])=="")
{
$objResponse->assign("ButtonFormEmailAdmin","disabled",false);
$objResponse->assign("ButtonFormEmailAdmin","value","Отправить>>");
$objResponse->alert("Пустая тема !!!");
return $objResponse;
}

if(trim($Id[frommailbody])=="")
{
$objResponse->assign("ButtonFormEmailAdmin","disabled",false);
$objResponse->assign("ButtonFormEmailAdmin","value", "Отправить>>");
$objResponse->alert("Вопрос не задан !!!");
return $objResponse;
}

$headers="From: <".$Id[frommail].">\n";
$headers.= "Subject: ".utf8in($Id[frommailtheme])."\n";
$headers.= "Content-type: text/plain; charset=\"windows-1251\"\n";
if(mail($Id[toemail],utf8in($Id[frommailtheme]),utf8in(
      $Id[frommailbody]),$headers))
{
$objResponse->assign("windowdop","style.display","none");
$objResponse->alert("Письмо отправлено !!!");
}
else
{
$objResponse->assign("ButtonFormEmailAdmin","disabled",false);
$objResponse->assign("ButtonFormEmailAdmin","value","Отправить>>");
$objResponse->alert("Ошибка отправки письма!!!");
}
return $objResponse;
}
?>
```

При нажатии на кнопку **Отправить** вызывается хайп-функция Go_Email_Admin, расположенная в файле prgcontacts/form_email_admin.php (листинг 4.44). Функция проверяет правильность заполнения полей и отправляет письмо на e-mail админи-

стратора. Для отправки используется функция `mail()`. При успешной отправке выдается сообщение (рис. 4.36) и окно исчезает (установка значения `none` для свойства `display` элемента с `id=windowdop`).

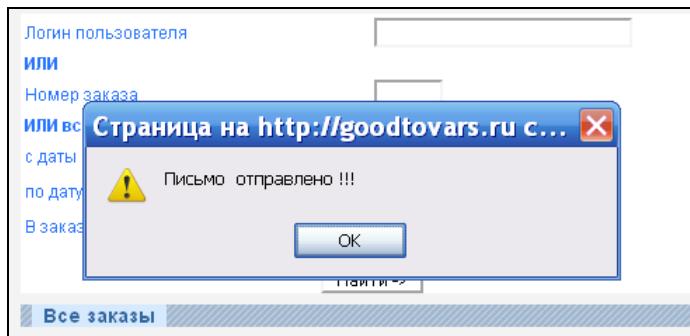


Рис. 4.36. Отправка письма на e-mail администратора из формы обратной связи

4.6.2. Обратная связь по ICQ

При нажатии на ссылку **385771293** вызывается хайп-функция `Form_ICQ_Admin`, расположенная в файле `prgcontacts/form_icq_admin.php` (листинг 4.45). Функция делает видимым окно (`div id=windowdop`) и создает в нем форму отправки сообщения на ICQ администратора (рис. 4.37).



Рис. 4.37. Форма обратной связи — отправка сообщения на ICQ администратора

Листинг 4.45

```
<?php
// Форма отправки icq админу с сайта
function Form_ICQ_Admin()
```

```
{  
$objResponse = new xajaxResponse();  
// подключение файла настроек  
require_once("my.php");  
// подключение базы данных  
require_once("mybaza.php");  
// сделать windowdor невидимым  
$objResponse->assign("windowdop","style.display","none");  
// создать форму  
$text1.=<a href='javascript:void(null);' onclick='  
    document.getElementById(\"windowdop\").style.display=  
    \"none\";return false;'>  
    <img src='img/delete.png' align=right></a>;  
$text1.=<center><b>Отправить вопрос<br>icq</b><br></center>;  
$text1.=<div id=VhodError></div>;  
$text1.=<form id='FormICQAdmin' action='javascript:void(null);'  
    onsubmit='xajax.$(\"ButtonFormICQAdmin\").disabled=true;  
    xajax.$(\"ButtonFormICQAdmin\").value=\"Подождите...\";  
    xajax_Go_ICQ_Admin(xajax.getFormValues(\"FormICQAdmin\"));  
    return false; '>;  
  
$text1.=<font color=black> Кому : Администратору сайта </font><input  
    type='hidden' name='toemail' value='".$ICQADMIN."'><br>;  
$text1.=<font color=black>От(Ваш ICQ) : </font><input type='text'  
    name='fromicq' value=''><br>;  
$text1.=<font color=black>Сообщение:<br></font><textarea  
    name='fromicqbody' cols=40 rows=7></textarea><br>;  
$text1.=<input id='ButtonFormICQAdmin' type='submit'  
    value='Отправить>>'>;  
$text1.=</center></form>;  
// сделать windowdor видимым  
$objResponse->script("document.getElementById('windowdop').style.display=  
    'block'");  
// выдать форму  
$objResponse->assign("windowdop","innerHTML",$text1);  
// windowdop в зону видимости  
$objResponse->script("document.getElementById('windowdop')  
    .scrollIntoView();");  
return $objResponse;  
}  
// Отправка icq на номер администратора  
function Go_ICQ_Admin($Id)  
{  
// подключение файла настроек  
require_once("my.php");
```

```

// подключение базы данных
require_once("mybaza.php");
$objResponse = new xajaxResponse();
// проверим - сообщение непустое
if(trim($Id[fromicqbody])=="")
{
    $objResponse->assign("ButtonFormICQAdmin","disabled",false);
    $objResponse->assign("ButtonFormICQAdmin","value","Отправить>>");
    $objResponse->alert("Вопрос не задан !!!");
    return $objResponse;
}
// подключение библиотеки WebIcqLite
include('WebIcqLite.class.php');
$icq = new WebIcqLite();
// соединение с сервером icq
if($icq->connect(ICQ, ICQPASS))
{
    // отправка сообщения
    if(!$icq->send_message(ICQADMIN, $Id[fromicqbody]))
    {
        // сделать активной кнопку
        $objResponse->assign("ButtonFormICQAdmin","disabled",false);
        $objResponse->assign("ButtonFormICQAdmin","value","Отправить>>");
        $objResponse->alert("Ошибка отправления сообщения icq !".
        $icq->error);
    }
    else
    {
        // сделать windowdor невидимым
        $objResponse->assign("windowdop","style.display","none");
        $objResponse->alert("Сообщение отправлено по icq!");
    }
    $icq->disconnect();
}
else
{
    $objResponse->assign("ButtonFormICQAdmin","disabled",false);
    $objResponse->assign("ButtonFormICQAdmin","value","Отправить>>");
    $objResponse->alert("Ошибка подключения к серверу icq !".$icq->error);
}
return $objResponse;
?>

```

При нажатии на кнопку **Отправить** вызывается хайп-функция Go_ICQ_Admin, расположенная в файле prgcontacts/form_icq_admin.php (листинг 4.45), которая про-

веряет правильность заполнения полей и отправляет письмо на ICQ администратора. Для отправки используется библиотека WebIcqLite (файл WebIcqLite.class.php). При успешной отправке выдается сообщение (рис. 4.38) и окно исчезает (установка значения none для свойства display элемента с id=windowdop).

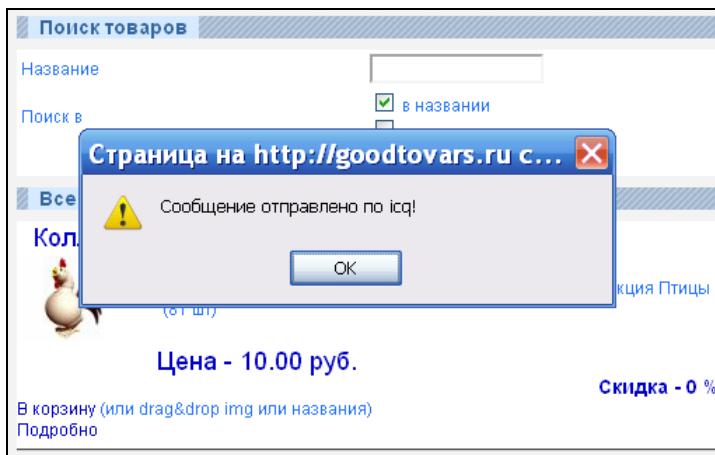


Рис. 4.38. Отправка сообщения на ICQ администратора из формы обратной связи

Заключение

Создание Web-сайтов требует от разработчиков немало опыта и навыков. При этом они должны постоянно следить за новшествами и быть в курсе современных технологий. Мы рассмотрели в книге один из самых современных и перспективных подходов к созданию сайтов без перезагрузки страниц — технологию AJAX. Вы получили возможность освоения новой технологии на примере создания достаточно большого проекта — интернет-магазина цифровых товаров. На прилагаемом компакт-диске вы найдете сайт, полностью готовый к размещению в Сети. Возможно, вы захотите внести в него какие-то изменения. Удобная архитектура сайта и навыки, полученные при чтении книги, позволят вам это сделать. Если у вас появятся вопросы, я всегда готов помочь. Пишите на электронную почту victoruni@km.ru и kmvnews@bk.ru или заходите в блог <http://goodtovars.ru/blog>, где мы будем разбирать уже готовые проекты — портал и биллинг-систему для хостинга на AJAX.

Буду рад, если изученный материал окажется полезен вам при написании собственных проектов.

Успехов вам и всего доброго!



ПРИЛОЖЕНИЯ



Приложение 1

Свойства стилей CSS

Свойство	Значение	Примечание
after	—	Используется для вывода желаемого контента после элемента, к которому он добавляется. Псевдоэлемент <code>after</code> работает совместно с атрибутом <code>content</code> : <code>after { content: "текст" }</code>
background	См. свойства <code>background-attachment</code> <code>background-color</code> <code>background-image</code> <code>background-position</code> <code>background-repeat</code>	Параметр позволяет установить одновременно до пяти атрибутов стиля фона. Значения могут идти в любом порядке, браузер сам определит, какое из них соответствует нужному атрибуту. Для более подробного ознакомления с аргументами,смотрите свойства каждого параметра отдельно
<code>background-attachment</code>	<code>fixed</code> — делает фоновое изображение элемента неподвижным; <code>scroll</code> — позволяет перемещаться фону вместе с содержимым	Параметр <code>background-attachment</code> устанавливает, будет ли прокручиваться фоновое изображение вместе с содержимым элемента. Изображение может быть зафиксировано и оставаться неподвижным либо перемещаться совместно с документом
<code>background-color</code>	<code>transparent</code> — прозрачный; <цвет> — значение цвета можно задавать тремя способами: 1) по его названию; 2) шестнадцатеричной константой, например #666999; 3) с помощью RGB	Устанавливает фоновый цвет элемента. Хотя этот параметр не наследует свойства своего родителя из-за того, что начальное значение цвета фона устанавливается прозрачным, он совпадает с фоном текущего элемента

Свойство	Значение	Примечание
background-image	none — нет фонового изображения; url ('путь к файлу')	Устанавливает фоновое изображение для элемента. Если одновременно для элемента задан цвет фона, он будет показан, пока фоновая картинка не загрузится полностью
background-position	top center bottom left center right <число> px <число> %	Задает начальное положение фонового изображения, установленного с помощью параметра <code>background-image</code> . Принимает два значения: положение по горизонтали и вертикали. Положение также можно задавать в процентах, пикселях или других единицах
background-repeat	no-repeat repeat repeat-x repeat-y	Определяет, как будет повторяться фоновое изображение, установленное с помощью параметра <code>background-image</code> , и по какой оси. Можно установить повторение рисунка только по горизонтали, по вертикали или в обе стороны
before	—	Применяется для отображения желаемого контента до элемента, к которому он добавляется. Псевдоэлемент <code>before</code> работает совместно с атрибутом <code>content</code> : <code>before { content: "текст" }</code>
border	См. свойства border-width border-style border-color	Параметр позволяет одновременно установить толщину, стиль и цвет рамки вокруг элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному атрибуту
border-bottom	См. свойства border-bottom-width border-bottom-style border-bottom-color	Параметр позволяет одновременно установить толщину, стиль и цвет границы внизу элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному атрибуту
border-bottom-color	<цвет> — цвет можно задавать 1) по его названию; 2) шестнадцатеричным значением;	Устанавливает цвет границы внизу элемента

	3) с помощью RGB	
Свойство	Значение	Примечание
border-bottom-style	dotted dashed solid double groove ridge inset outset	Устанавливает стиль границы внизу элемента
border-bottom-width	thin (2 пикселя); medium (4 пикселя); thick (6 пикселов); <число> px	Устанавливает толщину границы внизу элемента
border-collapse	collapse — отображается только одна линия между ячейками; separate — вокруг каждой ячейки отображается своя собственная рамка, в местах соприкосновения ячеек показываются сразу две линии	Устанавливает, как отображать границы вокруг ячеек таблицы. Этот параметр играет роль, когда для ячеек установлена рамка, тогда в месте стыка ячеек получится линия двойной толщины
border-color	<цвет> — цвет можно задавать: 1) по его названию; 2) шестнадцатеричным значением; 3) с помощью RGB	Устанавливает цвет границы на разных сторонах элемента. Параметр позволяет задать цвет границы сразу на всех сторонах элемента или определить цвет границы только на указанных сторонах. Разрешается использовать одно, два, три или четыре значения, разделяя их между собой пробелом
border-left	См. свойства border-left-width border-left-style border-left-color	Параметр позволяет одновременно установить толщину, стиль и цвет левой границы элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному атрибуту
border-left-color	<цвет> — цвет можно задавать: 1) по его названию; 2) шестнадцатеричным значением;	Устанавливает цвет левой границы элемента

	3) с помощью RGB	
Свойство	Значение	Примечание
border-left-style	Dotted dashed solid double groove ridge inset outset	Устанавливает стиль левой границы элемента
border-left-width	thin (2 пикселя); medium (4 пикселя); thick (6 пикселов); <число> px	Устанавливает толщину левой границы элемента
border-right	См. свойства border-right-width border-right-style border-right-color	Параметр позволяет одновременно установить толщину, стиль и цвет правой границы элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному атрибуту
border-right-color	<цвет> — цвет можно задавать: 1) по его названию; 2) шестнадцатеричным значением; 3) с помощью RGB	Устанавливает цвет правой границы элемента
border-right-style	dotted dashed solid double groove ridge inset outset	Устанавливает стиль правой границы элемента
border-right-width	thin (2 пикселя); medium (4 пикселя); thick (6 пикселов); <число> px	Устанавливает толщину правой границы элемента

Свойство	Значение	Примечание
border-spacing	<число> px	Задает расстояние между границами ячеек в таблице. Атрибут border-spacing не работает в случае, когда для таблицы установлен параметр border-collapse со значением collapse
border-style	dotted dashed solid double groove ridge inset outset	Устанавливает стиль рамки вокруг элемента. Допустимо задавать индивидуальные стили для разных сторон элемента
border-top	См. свойства border-top-width border-top-style border-top-color	Параметр позволяет одновременно установить толщину, стиль и цвет границы сверху элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному атрибуту
border-top-color	<цвет> — цвет можно задавать 1) по его названию; 2) шестнадцатеричным значением; 3) с помощью RGB	Устанавливает цвет границы сверху элемента
border-top-style	dotted dashed solid double groove ridge inset outset	Устанавливает стиль границы сверху элемента
border-top-width	thin (2 пикселя); medium (4 пикселя); thick (6 пикселов); <число> px	Устанавливает толщину границы сверху элемента

Свойство	Значение	Примечание
border-width	thin (2 пикселя); medium (4 пикселя); thick (6 пикселов); <число> px	Задает толщину границы одновременно на всех сторонах элемента или индивидуально для каждой стороны
bottom	<число> px <число> % auto	Устанавливает положение нижнего края содержимого элемента без учета толщины рамок и отступов. Отсчет координат зависит от параметра position, он обычно принимает значение relative (относительное положение) или absolute (абсолютное положение)
caption-side	top — располагает заголовок по верхнему краю таблицы; bottom — заголовок располагается под таблицей; right — заголовок размещается справа от таблицы; left — заголовок размещается слева от таблицы	Определяет положение заголовка таблицы, который задается с помощью тега <CAPTION> относительно самой таблицы. Параметр caption-side выводит заголовок до или после таблицы. Браузер Firefox также поддерживает расположение заголовка слева или справа от таблицы, но эти значения не входят в спецификацию CSS
clear	both — отменяет обтекание элемента одновременно с правого и левого края; left — отменяет обтекание с левого края элемента. При этом все другие элементы на этой стороне будут опущены вниз и располагаться под текущим элементом; right — отменяет обтекание с правой стороны элемента; none — отменяет действие данного свойства и обтекание элемента происходит, как задано с помощью параметра float или других настроек	Параметр устанавливает, с какой стороны элемента запрещено его обтекание другими элементами. Если установлено обтекание элемента с помощью параметра float, свойство clear отменяет его действие для указанных сторон

Свойство	Значение	Примечание
clip	rect(Y1, X1, Y2, X2) auto	Параметр определяет область позиционированного элемента, в которой будет показано его содержимое. Все, что не помещается в эту область, будет обрезано и становится невидимым. На данный момент единственная доступная форма области — прямоугольник. В качестве аргументов используется расстояние от края элемента до области вырезки, которое задается в единицах CSS — пикселях (px) и процентах (%)
color	<цвет> цвет можно задавать: 1) по его названию; 2) шестнадцатеричным значением; 3) с помощью RGB	Определяет цвет текста элемента
cursor	auto — вид курсора по умолчанию для текущего элемента; url — позволяет установить свой собственный курсор, указать путь к файлу, в котором указана форма курсора, в формате CUR или ANI; default crosshair help move pointer progress text wait n-resize ne-resize e-resize se-resize s-resize sw-resize w-resize nw-resize	Устанавливает форму курсора, когда он находится в пределах элемента. Вид курсора зависит от операционной системы и установленных параметров

Свойство	Значение	Примечание
display	block inline inline-block inline-table list-item none run-in table table-caption table-cell table-column table-column-group table-footer-group table-header-group table-row table-row-group	Многоцелевой атрибут, который определяет, как элемент должен быть показан в документе
float	left — выравнивает элемент по левому краю, а все остальные элементы, вроде текста, огибают его по правой стороне; right — выравнивает элемент по правому краю, а все остальные элементы огибают его по левой стороне; none — обтекание элемента не задается	Определяет, по какой стороне будет выравниваться элемент, при этом остальные элементы будут обтекать его с других сторон
font	См. свойства font-family font-size font-style font-weight font-variant	Параметр позволяет установить одновременно несколько атрибутов стиля шрифта. Значения могут идти в любом порядке, браузер сам определит, какое из них соответствует нужному атрибуту
font-family	Geneva Arial Helvetica Georgia Times New Roman и др.	Устанавливает семейство шрифта, которое будет использоваться для оформления текста содержимого. Список шрифтов может включать одно или несколько названий, разделенных запятой. Если в имени шрифта содержатся пробелы, например, Trebuchet MS, оно должно заключаться в одинарные или двойные кавычки. Когда браузер встречает первый шрифт в списке, он проверяет его наличие на компьютере пользователя. Если такого шрифта нет, берется следующее имя из списка и также анализируется на присутствие

Свойство	Значение	Примечание
font-size	<число> xx-small x-small small medium large x-large xx-large	Определяет размер шрифта элемента
font-style	normal — обычное начертание текста; italic — курсивное начертание; oblique — наклонный шрифт	Определяет начертание шрифта — обычное, курсивное или наклонное
font-variant	normal — не изменяет регистр символов, оставляя его по умолчанию; small-caps — модифицирует все строчные символы как заглавные уменьшенного размера	Определяет, как нужно представлять строчные буквы — делать их все прописными уменьшенного размера или оставить без изменений. Такой способ изменения символов называется капителью
font-weight	<число> (100-900) bold — полужирное; bolder — жирное; lighter — светлое; normal — нормальное начертание	Устанавливает насыщенность шрифта в диапазоне от 100 до 900 с шагом 100
height	<число> px <число> % auto	Устанавливает высоту блочных или заменяемых элементов (к ним, например, относится тег). Высота не включает толщину границ вокруг элемента, значение отступов и полей
left	<число> px <число> % auto	Для позиционированного элемента определяет расстояние от левого края родительского элемента, не включая отступ, поле и ширину рамки, до левого края дочернего элемента
letter-spacing	<число> px normal	Определяет интервал между символами в пределах элемента. Браузеры обычно устанавливают расстояние между символами, исходя из типа и вида шрифта, его размеров и настроек операционной системы. Чтобы изменить это значение, и применяется данный атрибут

Свойство	Значение	Примечание
line-height	normal <число> (множитель) <число> % <число> px	Устанавливает интерлиньяж (межстрочный интервал) текста, отсчет ведется от базовой линии шрифта
list-style	См. свойства list-style-image list-style-position list-style-type	Позволяет одновременно задать стиль маркера, его положение, а также изображение, которое будет использоваться в качестве маркера
list-style-image	none url('путь к файлу')	Устанавливает адрес изображения, которое служит в качестве маркера списка
list-style-position	outside — маркер вынесен за границу элемента списка; inside — маркер обтекается текстом	Определяет, как будет размещаться маркер относительно текста
list-style-type	none — устанавливает тип маркера, как у родительского элемента; для маркированного списка: circle disc square для нумерованного списка: decimal lower-alpha lower-roman upper-alpha upper-roman	Изменяет вид маркера для каждого элемента списка. Этот атрибут используется только в случае, когда значение свойства list-style-image установлено как none
margin	auto <число> px	Устанавливает величину отступа от каждого края элемента
margin-bottom	auto <число> px	Устанавливает величину отступа от нижнего края элемента
margin-left	auto <число> px	Устанавливает величину отступа от левого края элемента
margin-right	auto <число> px	Устанавливает величину отступа от правого края элемента
margin-top	auto <число> px	Устанавливает величину отступа от верхнего края элемента

Свойство	Значение	Примечание
max-width	<число> px <число> %	Устанавливает максимальную ширину элемента
min-width	<число> px <число> %	Устанавливает минимальную ширину элемента
opacity	<число> (0.0-1.0) 0 — полная прозрачность; 1 — непрозрачность	Определяет уровень прозрачности элемента Web-страницы. При частичной или полной прозрачности через элемент проступает фоновый рисунок или другие элементы, расположенные ниже полупрозрачного объекта
overflow	visible — отображается все содержание элемента, даже за пределами установленной высоты и ширины; hidden — отображается только область внутри элемента, остальное будет обрезано; scroll — всегда добавляются полосы прокрутки; auto — полосы прокрутки добавляются только при необходимости	Управляет отображением содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров
padding	<число> px <число> %	Устанавливает значение полей вокруг содержимого элемента. Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое
padding-bottom	<число> px <число> %	Устанавливает значение поля от нижнего края содержимого элемента
padding-left	<число> px <число> %	Устанавливает значение поля от левого края содержимого элемента
padding-right	<число> px <число> %	Устанавливает значение поля от правого края содержимого элемента
padding-top	<число> px <число> %	Устанавливает значение поля от верхнего края содержимого элемента

Свойство	Значение	Примечание
position	<p><code>absolute</code> — указывает, что элемент абсолютно позиционирован;</p> <p><code>fixed</code> — по своим свойствам это значение аналогично аргументу <code>absolute</code>, но в отличие от него привязывается к указанной параметрами <code>left</code>, <code>top</code>, <code>right</code> и <code>bottom</code> точке на экране и не меняет своего положения даже при пролистывании Web-страницы;</p> <p><code>relative</code> — положение элемента устанавливается относительно его исходного места;</p> <p><code>static</code> — элементы отображаются как обычно</p>	Устанавливает способ позиционирования элемента относительно окна браузера или других объектов на Web-странице
quotes	—	Устанавливает тип кавычек, который применяется в тексте документа: <code>quotes</code> : "левая кавычка" "правая кавычка"
right	<p><code>auto</code></p> <p><code><число> px</code></p> <p><code><число> %</code></p>	Для позиционированного элемента определяет расстояние от правого края родительского элемента, не включая отступ, поле и ширину рамки, до правого края дочернего элемента
table-layout	<p><code>auto</code> — браузер загружает всю таблицу, анализирует ее для определения размеров ячеек и только после этого отображает;</p> <p><code>fixed</code> — ширина колонок в этом случае определяется либо с помощью тега <code><COL></code>, либо вычисляется на основе первой строки</p>	Определяет, как браузер должен вычислять высоту и ширину ячеек таблицы, основываясь на ее содержимом
text-align	<p><code>center</code></p> <p><code>justify</code></p> <p><code>left</code></p> <p><code>right</code></p>	Определяет горизонтальное выравнивание текста в пределах элемента

Свойство	Значение	Примечание
text-decoration	blink — мигающий текст; line-through — создает перечеркнутый текст; overline — линия проходит над текстом (пример); underline — устанавливает подчеркнутый текст; none — отсутствие эффекта	Добавляет оформление текста в виде его подчеркивания, перечеркивания, линии над текстом и мигания. Одновременно можно применить более одного стиля, перечисляя значения через пробел
text-indent	<число> px <число> %	Устанавливает величину отступа первой строки блока текста (например, для параграфа <P>). Не оказывает воздействия на все остальные строки
text-transform	capitalize — каждое слово в предложении будет начинаться с заглавного символа; lowercase — все символы текста становятся строчными (нижний регистр); uppercase — все символы текста становятся прописными (верхний регистр)	Управляет преобразованием текста элемента в заглавные или прописные символы
top	auto <число> px <число> %	Для позиционированного элемента определяет расстояние от верхнего края родительского элемента, не включая отступ, поле и ширину рамки, до верхнего края дочернего элемента
vertical-align	baseline bottom middle super text-bottom text-top top	Выравнивает элемент по вертикали относительно своего родителя или окружающего текста
visibility	visible hidden collapse	Предназначен для отображения или скрытия элемента, включая рамку вокруг него и фон. При скрытии элемента, хотя он и становится не виден, место, которое элемент занимает, остается за ним

Свойство	Значение	Примечание
white-space	<p>normal — текст в окне браузера выводится как обычно, переносы строк устанавливаются автоматически;</p> <p>nowrap — переносы строк в коде HTML игнорируются, весь текст отображается одной строкой, вместе с тем, добавление тега
 переносит текст на новую строку;</p> <p>pre — текст показывается с учетом всех пробелов и переносов, как они были добавлены разработчиком в коде HTML. Если строка получается слишком длинной и не помещается в окне браузера, то будет добавлена горизонтальная полоса прокрутки</p>	Параметр устанавливает, как отображать пробелы между словами
width	auto <число> px <число> %	Устанавливает ширину блочных или заменяемых элементов (к ним, например, относится тег). Ширина не включает толщину границ вокруг элемента, значение отступов и полей
word-spacing	<число> px normal	Устанавливает интервал между словами
z-index	<число> auto	Любые позиционированные элементы на Web-странице могут накладываться друг на друга в определенном порядке, имитируя тем самым третье измерение, перпендикулярное экрану



Приложение 2

Описание компакт-диска

На прилагаемом компакт-диске находятся следующие папки:

- \Denwer_distr** — дистрибутивы базового пакета и расширений Денвера — пакета программ для создания и отладки сайтов на локальной машине;
- \book_primers** — файлы тренировочного стенда для изучения методов хаях и примеров из книги к главам 1 и 2. Здесь также находятся дампы баз данных к этим примерам;
- \magazin** — файлы корневого каталога и дамп базы данных сайта (интернет-магазина цифровых товаров). Для запуска сайта в сети Интернет необходимо скопировать файлы на хостинг, создать необходимые базы данных и поменять настройки в файле my.php;
- \sql** — файлы баз данных для сайта (тренировочного стенда) и примеров из книги для копирования в каталог установки Денвера (\usr\local\mysql5\data). При прямом копировании создаются две базы: magazin и book_examples;
- \xajax** — библиотека хаях, версия 0.5 standard;
- \jquery** — библиотека jQuery, версия 1.4.2 и Query UI — надстройки над JavaScript-библиотекой jQuery, версия 1.7.3;
- \pspad** — PSPad мощный текстовый редактор для Web-разработчиков и программистов (не требует установки, просто запустите файл pspad.exe). Очень рекомендую использовать этот редактор при разборе кода примеров и написании своих скриптов. Программа PSPad распространяется бесплатно (freeware), домашняя страница проекта <http://www.pspad.com>.

Предметный указатель

A

AJAX 44

C

CSS 9

D

DHTML 13

DOM 14

H

HTML 7

J

jQuery 92

M

MySQL 17

P

phpMyAdmin 25

Q

Query UI 104

S

Smarty 127

X

XAJAX 48

xajaxResponse 51

Г

Глобальные константы 61

Д

Денвер 31

Л

Листы стилей:

внешние 10

внутренние 10

М

Метод jQuery:

append 97

appendTo 97

attr 97

css 97

empty 98

html 98

prepend 97

prependTo 97

text 97

Метод Smarty:

assign 131

display 132

fetch 132

Метод xajaxResponse:

addEvent 56

alert 56

append 52

assign 51

call 56

clear 54

(окончание рубрики см. на стр. 424)

Метод jQuery (*окончание*):

create 53
createInput 54
includeScript 55
insert 53
insertAfter 53
insertInput 54
insertInputAfter 54
prepend 52
redirect 56
remove 53
removeHandler 55
replace 52
script 55

П**Плагин:**

Accordion 104
imageFlow 113
jCarousel 101
Tabs 105

Платежный сервис:

a1aggregator 190
ONPAY 245
Webmoney 242

С

Селектор 94