

## **1. Лекция: Основы Web-дизайна**

### **Контекст разработки Web**

Многие люди являются "авторами" страниц Web, немногие являются "разработчиками" сайтов Web. Возможно, вы также присоединитесь к сообществу разработчиков.

### **Создание страницы Web**

Сегодня создание страницы Web является не слишком трудной задачей. Многие стандартные программные пакеты персональных компьютеров обладают встроенными средствами для преобразования документов текстовых процессоров, электронных таблиц, баз данных и т.д. в специально кодированные документы, которые могут быть доступны в Web. Специальные пакеты для создания страниц Web, такие, как Microsoft FrontPage и Macromedia Dreamweaver, позволяют легко создавать страницы Web с помощью технологии буксировки. В большинстве таких случаев даже не нужно знать о существовании специального языка кодирования HTML (язык разметки гипертекста), который неявно все это обеспечивает.

Если вы знаете язык XHTML, то страницы Web можно создавать с помощью простого текстового редактора, получая в этом случае значительно больше контроля над их структурой и форматированием, чем это возможно с помощью методов буксировки. Кроме того, появляется возможность легко интегрировать существующий код XHTML, апплеты Java, встраиваемые модули мультимедиа и языки сценариев браузера, чтобы создать на странице некоторое взаимодействие с пользователем. Независимо от содержания или привлекательности страниц, их назначение обычно ограничено представлением интересного или информативного текста и графики для персонального потребления. Маловероятно, что кто-то будет заниматься задачей создания основной бизнес-системы с помощью HTML и нескольких подключаемых модулей.

### **Разработка Web**

"Разработка" Web, в противоположность "созданию" страниц Web, выходит далеко за пределы использования кодов разметки и нескольких подключаемых модулей или метода сценариев для создания привлекательных или информативных страниц Web. Этот термин относится к использованию специальных стратегий, инструментов и методов для создания страниц Web и сайтов Web, характеризующихся как трехуровневые, клиент/серверные системы обработки информации. Давайте рассмотрим эти термины более подробно, чтобы понять разнообразие задач, для которых разрабатываются страницы и сайты Web.

### **Системы обработки информации**

Технологии Web используются не только для создания персональных или рекламных сайтов Web, содержащих информативный, интересный или развлекательный материал для публичного потребления. Скорее они становятся важным средством поддержки фундаментальных "бизнес-процессов" современных организаций — поддерживающие операционные и управленческие функции. Технические инфраструктуры поддержки этих задач упрощенно делятся на три типа систем на основе Web, называемых системами интранет, интернет и экстранет.

### **Системы интранет**

Системы интранет являются частными, внутренними системами, помогающими выполнять повседневную обработку информации, управленческо-информационную и производственную деятельность организаций. Системы интранет на основе Web обслуживают стандартные внутренние функции бизнеса, оказывая тем самым влияние на основные организационные системы, такие, как бухгалтерский учет и финансовая отчетность, маркетинг и отдел продаж, системы закупок и сбыта, производственные системы, системы трудовых ресурсов и другие. Со временем системы интранет на основе Web станут основными техническими средствами, посредством которых будет осуществляться внутренняя деятельность организаций по выполнению бизнес-процессов.

### **Системы интернет**

Системы интернет являются публичными информационными системами. Они включают в себя публичные сайты, которые предоставляют новости, информацию, и развлечения; сайты электронной коммерции для маркетинга и продажи продуктов и услуг; правительственные сайты для информирования или обслуживания широкой публики; и образовательные сайты для предоставления локального и удаленного доступа к образованию и знаниям. Всем частям общества публичные системы интернет предоставляют товары, услуги и информацию посредством Всемирной паутины WWW и связанных с ней сетей и услуг.

### **Системы экстранет**

Системы экстранет являются системами бизнес-для-бизнеса (B2B), которые управляют электронным обменом данными (EDI) между деловыми предприятиями. Эти системы обеспечивают информационный поток между организациями – между компанией и ее поставщиками и между компанией и ее сбытовыми организациями – чтобы помочь в координации последовательности закупки, производства и распространения. Электронный обмен данными помогает исключить бумажный поток, сопровождающий бизнес-транзакции, используя технологии Web для пересылки электронных документов между компьютерами, а не между людьми.

Как системы на основе Web приложения EDI устраняют трудности передачи информации между различными программными и аппаратными

платформами с изначально различными информационными форматами и различными протоколами обмена информацией.

Web становится основным технологическим базисом, электронной магистралью для сбора информации, обработки и распространения во всех типах организаций – в коммерческих и финансовых предприятиях, образовательных учреждениях, правительственных агентствах, учреждениях здравоохранения, агентствах новостей и отрасли развлечений и в большинстве других формальных организаций, как больших, так и маленьких. Это всепроникающая технология для разработки систем работы с информацией во всех частях общества.

## **На основе Web**

Термин "на основе Web" относится к тому факту, что системы обработки информации полагаются на технологию Интернет, в частности, на так называемую Всемирную паутину (WWW). Поэтому системы на основе Web действуют в технологических рамках со следующими характеристиками.

Первое: системы действуют в публичных, а не в частных сетях данных. Они осуществляют коммуникацию через Интернет, т.е. через распространенные по всему миру, взаимосвязанные сети компьютеров, которые являются публично доступными.

Второе: коммуникационные сети основываются на открытых и публичных технических стандартах, таких, как архитектуры Ethernet, протоколы передачи TCP/IP и протоколы приложений HTTP и FTP. Они не являются частными или патентованными стандартами, но являются принципиально открытыми и свободными для публичного использования.

Третье: системы обработки на основе Web используют широко распространенное, часто бесплатное, программное обеспечение для разработки и работы. Деятельность по обработке происходит с помощью браузеров Web, а не специально написанного программного обеспечения для интерфейса пользователя и для внешнего сбора данных и обработки. Браузеры Microsoft Internet Explorer, Mozilla Firefox, Opera, Netscape Navigator и другие являются средством взаимодействия пользователей с системами обработки информации. Также широко распространенные компьютеры серверов Web выполняют основные функции бизнес-обработки, а серверы баз данных обеспечивают хранение информации, доступ к ней и извлечение.

Поэтому общедоступные, не являющиеся специализированными, не являющиеся патентованными оборудование и системы программного обеспечения предоставляют техническую среду для разработки систем обработки информации и для управления этой деятельностью.

## **Трехслойная, клиент/серверная архитектура**

Термин "клиент/сервер" относится к применению сетей на основе серверов для управления общим доступом к ресурсам и для распределения задач между аппаратными и программными компонентами. В клиент/серверных сетях на основе Web распределение задач обработки происходит в трех слоях, которые соответствуют трем основным компонентам оборудования/программного обеспечения системы.



Рисунок 1.1 - Аппаратные и программные слои трехслойной системы обработки информации

В первом слое (**Tier 1**) клиентский настольный ПК выполняет работу интерфейса пользователя системы; во втором слое (**Tier 2**) сервер Web выполняет основные функции системы по обработке; и в третьем слое (**Tier 3**) сервер базы данных, и в некоторых случаях медиа-сервер, осуществляет требуемые системе функции хранения и извлечения информации.

В свою очередь, каждый из трех аппаратных компонентов содержит соответствующее программное обеспечение. Клиентским программным обеспечением является браузер Web. Сервер Web выполняет сетевую операционную систему (NOS), такую, как Windows Server, Unix Server или Linux Server, и с помощью дополнительного программного обеспечения, например, Internet Information Server или Apache Web Server, реализует службы Интернет, — WWW, FTP, SMTP mail и другие. Сервер базы данных выполняет систему управления базой данных (DBMS), такую, как MySQL, Oracle, Access и другие популярные пакеты. Таким образом, отдельные компоненты выполняют отдельные задачи

обработки, которые интегрируются с помощью Web в законченную систему обработки информации.

Рассмотрим, например, посещение Web-сайта е-коммерции, например, Amazon.com. Браузер Web является интерфейсом с сайтом. В ответ на различные "входящие" запросы, которые вы отправляете при просмотре продаваемых товаров, создаются различные страницы "вывода". Запросы вводятся в систему через ссылки Web и посылаемые формы, ответы системы создают страницы HTML, передаваемые назад браузеру для вывода на экране. Браузер выполняет действия по вводу и выводу, необходимые для взаимодействия с сайтом.

За сценой на сервере Web решаются специальные задачи по обработке информации. Когда, например, делается запрос по поиску книги, выполняются программы поиска в базах данных для извлечения подходящих книг и для форматирования вывода для доставки в браузер Web. При просмотре корзины покупателя другие процедуры извлекают выбранные товары и вычисляют общую стоимость заказа. При оплате заказа исполняются специальные программы для соединения с системой проверки кредитной карты и банковскими системами, так что соответствующие счета дебетуются и кредитуются. Множество задач обработки, связанных с перемещением в сети и покупкой, происходят на серверах Web скрыто от пользователя, но они критически важны для осуществления покупки и для осуществления бизнес-транзакций, которые с этим связаны.

Большая часть информации, которая собирается и генерируется во время покупки, хранится в базах данных, которые находятся на отдельных серверах баз данных. Вся информация, которая выводится на экран, извлекается из таблиц базы данных. Выбранные товары хранятся в таблицах базы данных. Практически каждый фрагмент информации о просматриваемых продуктах и транзакциях при покупке сохраняется в больших базах данных в самой системе е-коммерции или в связанных базах данных, которые находятся в центре окружающих ее систем бухгалтерского учета, закупок и дистрибуции.

Даже в самых маленьких коммерческих системах на основе Web присутствуют такие же функции. Браузер Web предоставляет интерфейс пользователя с системой, специальные страницы обрабатывают бизнес транзакции, а одна или несколько баз данных поддерживают информацию, перемещающуюся в системе. Главное состоит в том, что в системах на основе Web любого размера существуют три основных слоя функциональности. Поэтому, с точки зрения разработчика Web, задача состоит в создании трех отдельных компонентов – интерфейса пользователя, процедур обработки бизнес-операций, и компонентов поддержки базы данных — и последующей интеграции в полностью функциональную систему обработки информации.

## **Навыки разработки Web**

Разработка Web относится к использованию технологий Web для создания клиентских и серверных компонентов обработки, к интеграции их в качестве приложений в системах обработки интранет, интернет или экстранет, и к развертыванию их в Web для реализации частной и публичной деятельности организаций. Набор навыков для реализации этих задач простирается далеко за пределы способности сохранить страницы Web из программы текстового процессора, за пределы создания простого сайта Web с помощью буксировки в программном пакете настольного компьютера или даже за пределы жестко закодированных с помощью XHTML страниц с вкраплениями подключаемых модулей.

При окончательном анализе разработчику Web необходимо проникновение в сущность операционных и управленческих процессов организации, понимание того, каким образом производственные процессы создаются и опираются на информационные потоки при производстве товаров и услуг, и способность абстрагироваться и моделировать эти бизнес системы на доступном оборудовании и программных технологиях, а также навыки использования этих технологий для создания систем на основе Web, которые реализуют эти модели.

Один учебник не может вместить все это. Он может, однако, предоставить основные знания и навыки для решения существующих проблем и использования возможностей, сопровождающих системы обработки на основе Web. Он охватывает основные технические средства для интеграции клиентского и серверного оборудования и программного обеспечения для создания систем по сбору, обработке, управлению и распространению информационного содержания, которое оживляет современные организации. Попутно вы получите надежное понимание той критической роли, которую системы на основе Web могут играть в создании операционного и управленческого успеха различных организаций.

## **Модели систем Web**

Исторически Всемирная паутина WWW функционировала просто как "система доставки информации". Люди привыкли использовать ее для сбора информации по всевозможным вопросам, для которых миллионы сайтов Web предоставили доступ. Однако со временем Web стала чем-то большим, чем просто электронной библиотекой информации. Она стала платформой коммуникации, информации и транзакций, на которой реализуется экономическая, социальная, политическая, образовательная и культурная деятельность.

## **Модель доставки информации**

При функционировании в качестве системы доставки информации деятельность по разработке Web — достаточно простая и прямолинейная. Прежде всего, информационное содержимое вводится в документ, который со временем станет страницей Web. Это содержимое окружается специальными кодами компоновки и форматирования Языка разметки гипертекста (HTML) — в последнее время Расширяемого языка разметки гипертекста (XHTML) – для управления его структурой и представлением в браузере Web.

Затем документ сохраняют на компьютере сервера Web для ожидания публичного доступа. Пользователи обращаются к документу, вводя в окне своего браузера адрес Web-документа. Этот адрес, называемый URL, или Единообразный локатор ресурса, определяет сайт, где хранится страница, и расположение ее каталога на сервере Web. Этот сервер, в свою очередь, извлекает страницу и посылает ее браузеру, который интерпретирует код HTML и выводит документ на экране компьютера.

Существуют определенные последствия построения доступа к Web на модели доставки информации и в следовании традиционному процессу разработки Web. Прежде всего, информационное содержимое страницы Web "фиксируется" или "замораживается" в определенном месте. Оно становится встроенным и тесно связанным с кодами форматирования XHTML, которые его окружают. В связи с этим становится трудно изменять содержимое страницы, не переписывая и не редактируя его форматы представления. Поэтому затрудняется сохранение актуальности страниц, особенно если содержимое постоянно изменяется.

В то самое время авторам страниц Web зачастую необходимо быть знакомым с кодированием XHTML. Даже при использовании визуальных инструментов, таких, как FrontPage или Dreamweaver, автору может понадобиться специалист по кодированию, чтобы страница выглядела требуемым образом. "Эксперту" Web часто также бывает необходимо работать в тесном контакте с поставщиком контента, обеспечивая технические навыки для сопровождения страниц.

Для пользователей также имеются ограниченные возможности взаимодействия с традиционными страницами Web. Пользователь часто выступает в роли пассивного читателя контента, для которого сервер Web действует в качестве простого электронного "переворачивателя страниц". Поэтому сайт Web, создаваемый вокруг модели доставки информации, может стать статическим, пассивным хранилищем устаревшей информации. Страница Web рискует стать историческим архивом, а не своевременным, быстро реагирующим источником точной, самой свежей информации.

## Модель обработки информации

Чтобы преодолеть это статическое, пассивное использование Web, возникает необходимость рассматривать Web не просто как систему доставки информации, но как полнофункциональную систему обработки информации. Это означает, что сама система Web и составляющие ее сайты и страницы необходимо воспринимать как механизмы для выполнения полного набора действий по вводу, обработке, выводу и хранению, требуемых для создания динамического, активного контента, – короче, для обеспечения основных функций системы обработки информации.

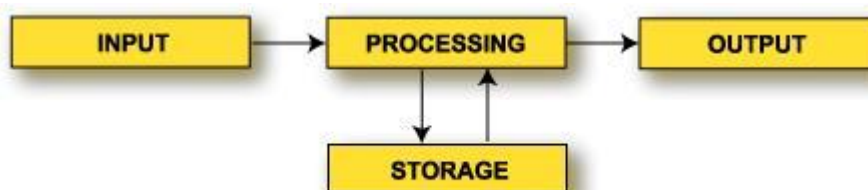


Рисунок 1.2 - Функции системы обработки информации

В модели информационной обработки четыре базовые функции ввода, обработки, вывода и хранения имеют специфическое значение.

- Функция ввода позволяет пользователям взаимодействовать с системой, запрашивая параметры обработки, управляя информационным доступом и определяя методы доставки. Кроме того, пользователь может стать источником данных, которые обрабатывает система и которые она поддерживает в своих репозиториях хранимой информации.
- Функция обработки относится к деятельности по манипуляции данными и логике обработки, необходимых для выполнения работы системы. Этот термин предполагает, что система может "программироваться" для выполнения арифметических и логических операций, необходимых для манипуляции данными ввода и для создания выводимой информации.
- Функция вывода доставляет результаты обработки пользователю в правильном, своевременном и соответствующем образом форматированном виде.
- Функция хранения гарантирует продолжительность существования и целостность обрабатываемой информации, поддерживая ее в течение длительного периода времени и позволяя добавлять, изменять или удалять систематическим образом. В конечном счете, хранимая информация становится основным контентом страниц Web, отражая самую современную и точную информацию, появляющуюся на этих страницах.

С точки зрения обработки информации сама сеть Web функционирует как гигантская открытая компьютерная система, и фактически такой и



является. Деятельность по обработке информации происходит на различных аппаратных и программных компонентах, расположенных в одном месте или разбросанных по всему миру.

При принятии модели обработки информации можно начинать применять технологию Web для создания сайтов Web, которые являются действительно динамичными, интерактивными и современными – сайты Web, на которых информационный контент всегда самый современный, персонализирован в соответствии с потребностями пользователя, автоматически изменяется в ответ на запросы пользователя, и когда изменяется сама информация, пользователь может взаимодействовать со страницей Web, добавляя или изменяя информацию в системе. Дополнительное преимущество состоит в том, что сайты можно создавать таким образом, что не потребуется постоянно переписывать или переформатировать страницы Web. Сами страницы изменяются динамически, отражая изменение информации или изменение предпочтений пользователей.

### **Присваивание функций компонентам**

Возвращаясь к понятию трехслойной, клиент/серверной системы, посмотрим, как аппаратные компоненты, программные компоненты, и функции обработки связаны друг с другом с точки зрения обработки информации. Вспомните, что аппаратные компоненты в трехслойной среде состоят из компьютера клиента Web (настольного ПК), компьютера сервера Web, и сервера базы данных. Каждый из этих трех аппаратных компонентов выполняет соответствующее программное обеспечение. Теперь можно отобразить функции обработки информации на эти три слоя оборудования/программ.



Рисунок 1.3 - Функции системы обработки информации, отображенные в трехслойную систему клиент/сервер

Все функции ввода и вывода попадают в основном в клиентскую машину Web. Такая деятельность интерфейса пользователя, как ввод данных, проверка данных, управление обработкой и форматирование вывода, выполняются на клиенте Web.

Основная деятельность по обработке информации падает на сервер Web, называемый иногда сервером приложений Web. Здесь программируются арифметические и логические процедуры для выполнения задач системы по бизнес обработке.

Наконец, хранение данных и функции доступа выполняются на сервере базы данных. Этот компонент управляет хранением информации и функциями извлечения, необходимыми для выполнения бизнес-обработки, и позволяет осуществлять долгосрочное обслуживание

хранимой информации посредством добавления, изменения и удаления файлов и баз данных.

Как можно видеть на иллюстрации, системные функции "передаются" различным отдельным компонентам, хотя все они действуют совместно, как интегрированные системы различных видов деятельности. Основная идея заключается в том, что специализированные компоненты выполняют специализированную работу, для которой они лучше всего подходят. Справедливо также то, что системные функции "не привязаны к месту". То есть деятельность ввода, обработки, вывода и хранения могут происходить там, где расположены компоненты. Они могут быть заключены в одной машине на одном рабочем столе, распределены между двумя или несколькими машинами в отделе или компании либо широко разбросаны по всему земному шару. Во всех этих случаях используемые технологии и методы являются практически одинаковыми, делая достаточно рутинной разработку приложений Web для любой физической или географической среды, с которой столкнется разработчик.

## **Разработка приложений Web**

Рассмотрение Web как трехслойной, клиент/серверной системы обработки информации имеет важные последствия для разработки приложений Web. Вчерашний "создатель страницы Web" становится сегодня "разработчиком системы Web". Теперь недостаточно иметь пакет разработчика, состоящий из редактора WYSIWYG и базовых навыков XHTML. Необходимо стать широко образованным в более разнообразном множестве технологий. Внизу на следующей расширенной схеме трехслойной системы указаны некоторые навыки и инструменты, необходимые для проектирования и программирования систем на основе Web, которые обслуживают деятельность по обработке информации.

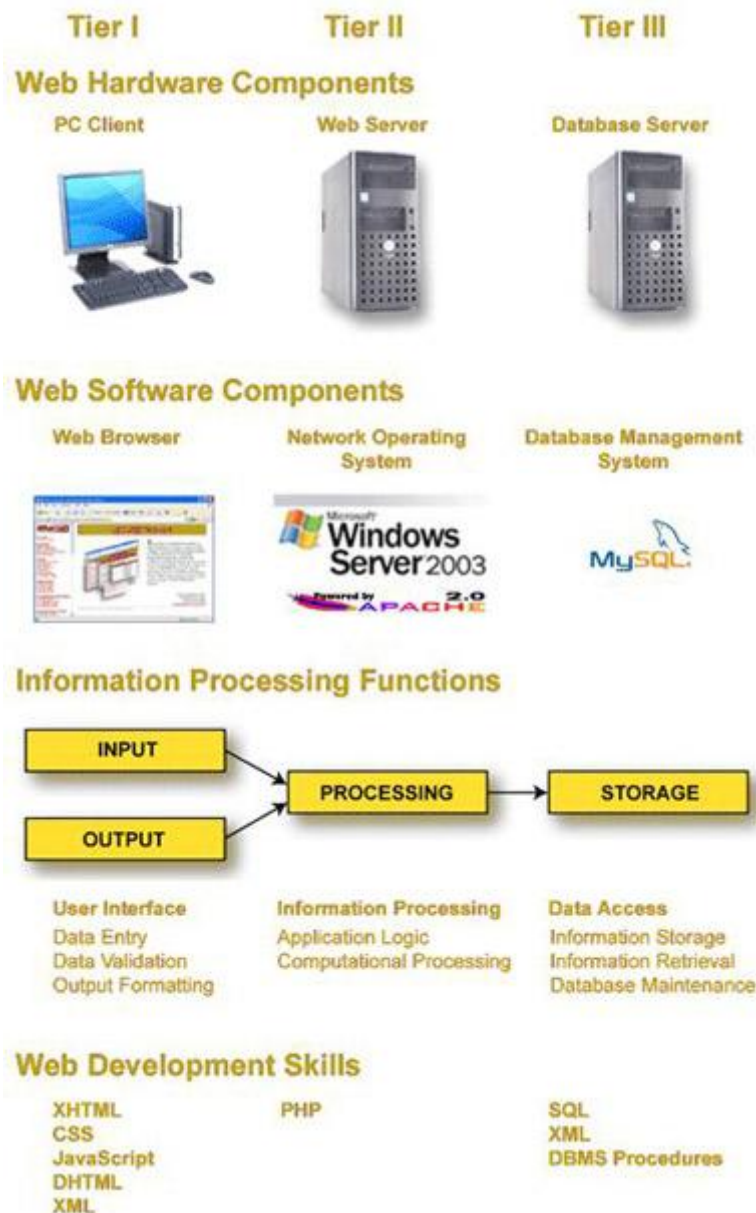


Рисунок 1.4 - Навыки разработки Web, требуемые для создания трехслойных клиент/серверных приложений на основе Web

## Системный ввод и вывод

Так как функция интерфейса пользователя осуществляется через браузер Web, выполняющийся на клиентском ПК, необходимо использовать инструменты разработки приложений, которые позволяют программировать браузер для выполнения задач форматирования вывода, ввода данных и проверки данных. Для этого требуются, конечно, языки разметки, такие, как XHTML и CSS (Каскадные таблицы стилей), для структуризации и представления системного ввода и вывода. Растет важность расширений языков разметки, например, DHTML (Dynamic HTML) для взаимодействия пользователей со страницей Web и XML (Расширяемый язык разметки) для представления структур

данных, которые доставляются сервером для обработки в браузере. Также основным языком программирования для браузера является JavaScript, который используется для манипуляции языками разметки и структурами данных для выполнения задач браузера по обработке.

## **Системная обработка**

На стороне сервера Web необходимо иметь возможность писать приложения для выполнения основных задач системы по обработке. Для кодирования этих процедур используются серверные языки, такие, как PHP. С одной стороны, эти языки применяются как полноценные языки программирования для кодирования арифметических и логических операций обработки, с другой — они используют встроенные компоненты серверной обработки для выполнения основных и вспомогательных задач системы. PHP является широко распространенным языком сценариев общего назначения, который особенно хорошо подходит для разработки Web и может встраиваться в XHTML.

## **Управление базой данных**

На стороне сервера базы данных такие языки, как SQL (Язык структурированных запросов), выполняют функции сохранения данных, их обслуживания и доступа. Кроме того, языки программирования базы данных используются для кодирования командных процедур, которые являются функциями обработки, хранимыми в базах данных, для извлечения, обновления и создания отчетов о содержимом баз данных. Серверные языки могут вызывать эти встроенные процедуры баз данных для выполнения соответствующей обработки, а не кодировать их непосредственно. Все большую важность в хранении данных и электронном обмене данными приобретают структуры данных XML.

Иногда на разработку Web смотрят упрощенно с чисто технической точки зрения, забывая, что это является также разработкой системы. Разработчик должен понимать организационные структуры и процессы. Прежде всего, система на основе Web является бизнес-процессом. Если не понимать суть процессов, то маловероятно, что можно будет разработать системы для их реализации или поддержки. Разработка Web является также реализацией некоторой интеграции. Задача состоит в том, чтобы объединить совокупность оборудования, программного обеспечения, людей и процедур для выполнения некоторой деятельности. Поэтому системный подход является критически важным для соединения всех частей вместе во что-то функциональное, продуктивное, экономичное и дружелюбное. Наконец, разработка Web является в большой степени творческим предприятием. Очень часто работа состоит в разработке чего-то нового там, где ничего перед этим не существовало. Вместо следования директивам хорошо продуманных планов, ваше воображение создает эти планы, а художник внутри вас их воплощает. Управляемый полет фантазии хорошо служит разработке Web.

Разработчику Web требуется поэтому творческое воображение, организаторское чутье и обширный набор навыков для создания приложений Web, которые обслуживают потребности информационной обработки. Разработчик не обязан быть экспертом высокого уровня во всех языках и программных инструментах; но необходимо иметь хорошее представление об их использовании.

## **Программное обеспечение учебника**

В этом учебнике представлено основное подмножество инструментов, необходимых для создания динамических, интерактивных сайтов Web. На стороне браузера применяются и описываются XHTML, CSS, JavaScript для редактирования данных, и динамический HTML. В качестве языка сценариев сервера используется PHP. Также описываются встроенные функции PHP, необходимые для выполнения функций ввода, обработки, вывода и хранения. На стороне базы данных представлены примеры для Microsoft Access, и описаны основные принципы использования языка SQL.

Даже при отсутствии достаточного доступа к этим технологиям, необходимо уметь следовать инструкциям и начать создавать свои собственные динамические приложения Web. Необходимо, конечно, разбираться в программировании: разработчик Web, прежде всего, является программистом. Предполагается, что читатель хорошо знаком с Visual Basic. Необходимо также иметь навыки работы с XHTML и быть знакомым с каскадными таблицами стилей.

## **Язык PHP**



PHP означает Препроцессор гипертекста PHP. Это серверный язык программирования, созданный специально для динамических страниц Web. Язык был первоначально разработан в 1994 г. Расмусом Лерддорфом и был с тех пор расширен, чтобы стать одним из наиболее популярных языков сценариев WWW. Согласно статистике Netcraft в 2005 г. PHP использовался более чем на 23000000 доменах. Подобно другим типам серверных языков, таких, как ASP, ASP.NET и JSP, код PHP обрабатывается на сервере Web и создает код XHTML или другой вывод, который можно увидеть в браузере. В отличие от других серверных языков, PHP является продуктом с открытым исходным кодом — это означает, что каждый имеет доступ к исходному коду и может использовать, изменять и распространять его полностью бесплатно.

Текущая версия PHP, рассматриваемая в учебнике, имеет номер 5. Этот учебник, несомненно, не является полным изложением языка PHP. Он

должен только продемонстрировать некоторые наиболее широко используемые свойства и приложения PHP.

Систему PHP5 можно применять практически с любым типом операционной системы и сервера Web. Однако, чтобы сценарии PHP были обработаны, должен быть установлен интерпретатор PHP. Это программное обеспечение доступно в двух формах – полный исходный код и исполняемые двоичные файлы. Большинство систем Linux поставляются с исходным кодом PHP. Для систем, отличных от Unix/Linux, двоичные файлы можно загрузить по адресу <http://www.php.net/downloads.php>.

Дополнительную информацию о PHP можно найти на сайте <http://www.php.net>.

## **2. Лекция: Основная структура документа**

### **Соединение XHTML и PHP**

Код PHP обычно объединяется с тегами XHTML. PHP является встраиваемым языком — это означает, что можно перемещаться между чистым кодом HTML и PHP, не жертвуя возможностью чтения текста.

Чтобы встроить код PHP в XHTML, PHP должен задаваться обособленно, с помощью начального и конечного тегов PHP. Теги PHP говорят серверу Web, где начинается и заканчивается код PHP. Анализатор PHP распознает три варианта начального и конечного тегов.

- Стиль XML
- `<?php`
- Блок кода PHP
- `?>`

Первый вариант тегов PHP называется тегами в стиле XML и является предпочтительным стилем. Он работает в документах Расширяемого языка разметки (XML). Этот метод должен использоваться при соединении PHP с документами XML и XHTML. Примеры в этом учебнике применяют этот формат тегов XML.

- Сокращенный стиль
- `<?`
- Блок кода PHP
- `?>`

Сокращенный стиль является самым простым, однако, он не рекомендуется, так как вступает в противоречие с объявлениями документов XML.

- Стиль сценария (script)
- `<script language="php">`

- Блок кода PHP
- </script>

Этот стиль использует самую длинную запись и похож на стиль тегов, применяемых с JavaScript. Этот стиль является предпочтительным при использовании редактора HTML, который не распознает другие стили тегов. Так как большинство новых редакторов XHTML распознают стиль тегов XML, то использование этого стиля не рекомендуется.

Блоки сценария могут размещаться в любом месте документа XHTML, в том месте, где сценарий создает и показывает свой вывод. Следующий пример страницы XHTML иллюстрирует использование трех форматов тегов сценария.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>Страница Web </title>
</head>
<body>

<p>
<?php echo "Это базовый документ PHP";?>
</p>

<p>
<? print "PHP - это здорово!";?>
</p>

<p>

<script language="php">
$myvar = "Hello World! ";
echo $myvar;
</script>

</p>

</body>
</html>
```

В предыдущем примере три блока PHP включены в документ XHTML. Первый блок использует открывающий и закрывающий теги <?php ... ?>. Сегмент кода использует оператор PHP `echo` для вывода строки "Это основной документ PHP" в окне браузера.



Второй блок применяет теги `<? ... ?>` для пометки начала и конца кода PHP. Этот раздел применяет оператор PHP `print` (другое имя оператора `echo`) для вывода на экране текста "PHP – это здорово!".

Наконец, третий блок использует блок сценария `<script language="php"> ... </script>` для определения начала и конца кода PHP. В коде строка "Hello World" присваивается переменной `$myvar`, а оператор `echo` выводит значение `$myvar` в окне браузера.

Это базовая страница PHP.

PHP – это здорово!

Hello World

Пример показанного выше кода включает теги XHTML, теги PHP, операторы PHP и разделители. Когда пользователь запрашивает страницу PHP, сервер обрабатывает весь код PHP. Когда страница PHP просматривается в окне браузера, выводится только текст между открывающим и закрывающим тегами XHTML или PHP. Никакой реальный код PHP не виден при просмотре исходного кода в окне браузера. Причина в том, что интерпретатор PHP выполняет сценарий на сервере и заменяет код результатом вывода работы сценария. Только этот вывод передается браузеру. Это одна из характеристик, которая делает PHP серверным языком сценариев, в отличие от JavaScript, языка сценариев клиента.

## Вывод контента

PHP содержит два основных оператора для вывода текста в браузере Web: `echo` и `print`.

Оба оператора, `echo` и `print`, кодируются между открывающим и закрывающим тегами блока кода PHP и могут находиться в любом месте в документах XHTML.

Операторы `echo` и `print` используют следующий формат:

`echo` – используется для вывода одной или нескольких строк.

```
echo "Выводимый текст";
```

`print` – используется для вывода строки. В некоторых случаях оператор `print` предлагает большую функциональность, чем оператор `echo`. Это будет рассмотрено далее в учебнике. Пока `print` можно считать другим именем оператора `echo`.

```
print " Выводимый текст"
```

Следующие примеры демонстрируют использование и размещение команд `echo` и `print` в документе XHTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>Страница Web</title>
</head>
<body>

<p>

<?php
echo "Это базовый документ PHP";
?>

</p>

</body>
</html>
```

В большинстве случаев необходимо выводить целые параграфы в окне браузера или создавать переносы строк при выводе контента. По умолчанию операторы `echo` и `print` не создают автоматические переносы строк, необходимо использовать тег `<p>` или `<br>` для создания параграфов или переносов строк. Разделители, создаваемые в редакторе XHTML с помощью возврата каретки, пробелов и табуляции, игнорируются процессором PHP.

В следующем примере тег параграфа XHTML включается в оператор PHP `echo`. В PHP теги XHTML можно применять в операторах `print` и `echo` для форматирования вывода. В этих случаях вывод необходимо заключать в двойные кавычки (`"`), чтобы гарантировать, что браузер не интерпретирует тег буквально и не выводит его как часть строки вывода.

```
echo "<p>Параграф 1</p>";
echo "<p>Параграф 2</p>";
```

Без использования тега параграфа XHTML предыдущие операторы `echo` будут выводить контент в следующем виде:

Параграф 1 Параграф 2

При включении тегов параграфов операторы выводятся как два отдельных параграфа.

Параграф 1

## Терминатор инструкции

Каждая строка кода PHP должна завершаться терминатором инструкции (признаком конца), в качестве которого используется точка с запятой (;). Терминатор инструкции применяется для отделения одного множества инструкций от другого.

Отсутствие терминатора инструкции может приводить к ошибкам в работе интерпретатора PHP и выводу ошибок в окне браузера.

Следующий фрагмент кода показывает распространенную ошибку отсутствия терминатора инструкции.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>Страница Web </title>
</head>
<body>

<p>

<?php

echo "Это строка порождает ошибку"
echo "В предыдущей строке отсутствует терминатор инструкции";

?>

</p>
</body>
</html>
```

В этом примере в первом операторе `echo` пропущен терминатор инструкции. Так как интерпретатор PHP не может определить конец первого оператора `echo` и начало второго, то происходит ошибка. В зависимости от настроек Обработки ошибок (Error Handling) интерпретатора, будет выводиться сообщение об ошибке или браузер выведет просто пустую страницу.

Далее показано типичное сообщение об ошибке, которое выводится, когда пропущен терминатор инструкции:

```
Parse error: syntax error, unexpected T_PRINT in
C:\ApacheRoot\test.php on line 11
Ошибка при разборе: синтаксическая ошибка, непредвиденный
T_PRINT в ... в строке 11
```

Как можно видеть, сообщения об ошибках PHP достаточно загадочны по своей природе. Отсутствие терминатора инструкции является распространенной ошибкой среди новичков PHP. Со временем появится привычка проверять каждую строку на наличие терминатора инструкции.

## Комментарии в коде

Комментарии применяются в PHP для записи собственных замечаний во время процесса разработки кода. Такие комментарии могут определять назначение сегмента кода или их можно использовать для исключения блоков кода во время тестирования и отладки сценариев.

Синтаксический анализатор PHP игнорирует комментарии. Комментарии в PHP можно определить одним из следующих способов:

// — простой комментарий PHP;

# — альтернативный простой комментарий PHP;

/\*...\*/ — многострочные блоки комментариев.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml11-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
```

```
<head>
```

```
    <title>Страница Web </title>
```

```
</head>
```

```
<body>
```

```
<p>
```

```
<?php
```

```
// Простой комментарий PHP
```

```
# Другой тип простого комментария PHP
```

```
/* Многострочный блок комментария PHP
```

```
Он может занимать любое необходимое
количество строк */
```

```
?>
```

```
</p>
```

```
</body>
```

```
</html>
```

### 3. Лекция: Данные PHP

#### Скалярные переменные

Переменные являются временным местом хранения, используемым для представления значений в сценарии PHP. В PHP имеется два основных типа переменных: скалярные и массивы. Скалярные переменные содержат только одно значение в данный момент времени, а переменные массивы — список значений. Переменные массивы обсуждаются в следующем разделе. Скалярные переменные PHP содержат значения следующих типов.

Целые – целые числа или числа без десятичной точки (1, 999, 325812841).

Числа с плавающей точкой – числа, содержащие десятичную точку (1.11, 2.5, .44).

Строки – текстовая или числовая информация. Строковые данные всегда определяются с помощью кавычек ("Hello World", "478-477-5555").

Булевы значения – используются для значений `true` (истина) или `false` (ложь).

Имена переменных PHP всех типов начинаются со знака "\$". Имена переменных могут содержать буквы, числа, и символ подчеркивания (`_`); они не могут, однако, начинаться с цифры. В PHP имена переменных различают регистр символов. Следующие переменные в PHP интерпретируются как две различные переменные.

```
$myvar  
$MYVAR
```

Допустимые имена переменных:

```
$myvar  
$F_Name  
$address1  
$my_string_variable
```

Недопустимые имена переменных:

```
Myvar  
$1stvar  
$&62##
```

Скалярным переменным PHP присваивают значения в следующем формате:

```
$username = "jdoe"
```

```
$first_name = "John"
$Last_Name = "Doe"
```

Переменная `username` содержит значение `jdoe`.

## Вывод переменных

Следующий фрагмент кода демонстрирует, как объявить скалярную переменную, присвоить скалярной переменной значение и вывести результаты в окне браузера:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml11-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>Страница Web</title>
</head>
<body>

<p>

<?php

$string_var = "Моя программа PHP";
$integer_var = 500;
$float_var = 2.25;

echo $string_var;
echo $integer_var;
echo $float_var;

?>

</p>
</body>
</html>
```

Переменные массивы PHP можно создавать и присваивать им значения с помощью конструкции `array()` или явным образом.

Переменную можно соединять с другими переменными или тегами XHTML с помощью оператора PHP — точки (`.`). В предыдущем блоке кода значения переменных выводятся в следующем формате:

```
Моя программа PHP5002.25
```

Чтобы создать возврат каретки или перенос строки, можно присоединить тег XHTML `<br/>` в конце каждой переменной:

```

<?php

$string_var = "My PHP program" . "<br/>";
$integer_var = 500 . "<br/>";
$float_var = 2.25;

echo $string_var;
echo $integer_var;
echo $float_var;

?>

```

Теперь после каждой переменной вставляется перенос строки, что приводит к выводу каждого значения на отдельной строке.

```

My PHP Program
500
2.25

```

## Соединение переменных

Оператор точки можно использовать также для соединения строк и переменных:

**Сообщение** — The user's name is John Doe — **выводится в окне браузера.**

Строка "The user's name is " соединяется со значением \$fname (John), за которым следует пробел " ", и значением \$lname (Doe).

```

<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>A Web Page</title>
</head>
<body>

<p>

<?php

$fname = "John";
$lname = "Doe";

echo "The user\'s name is " . $fname . " " . $lname;

?>

```

```
</p>
</body>
</html>
The user's name is John Doe
```

## Интерполяция

PHP поддерживает также процесс, называемый интерполяцией – замену переменной в строке ее содержимым. Вместо соединения переменных и литералов, их можно объединять внутри двойных кавычек ("").

Интерполяция является свойством только двойных кавычек. Переменные и литералы нельзя объединить внутри одиночных кавычек. При использовании двойных кавычек значение переменной выводится вместе с литералом. При использовании одиночных кавычек выводится "буквально" имя переменной вместе с остальной строкой. Следующий пример иллюстрирует свойство интерполяции PHP.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>A Web Page</title>
</head>
<body>

<p>

<?php

$fname = "John";
$lname = "Doe";

echo "The user\'s name is $fname $lname";

?>

</p>
</body>
</html>
```

Этот код создает такой же вывод, как и предыдущий пример. Здесь переменные объединяются с помощью литеральной строки, заключенной в двойные кавычки. Соединение (конкатенация) не требуется.

## Форматирование вывода валюты

Кроме вывода стандартного текста можно применять для вывода форматированного текста вариант конструкции `print` с именем `sprintf`. Оператор требует задания формирующей строки и значения для форматирования.



`printf("%01.2f", $var)` - выводит значение '\$var' как валюту.

Оператор `printf` показан ниже:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
  <title>A Web Page</title>
</head>
<body>

<p>
<?php
$amount = 35;
$tax = 2.50;
$total = $amount + $tax;
echo "$" . printf("%01.2f", $total);
?>
</p>

</body>
</html>
```

Вывод валюты показан ниже:

\$37.50

## Переменные массивы

В то время как скалярная переменная PHP хранит одно значение, переменную массива можно использовать для хранения множества или последовательности значений. Система PHP поддерживает массивы с числовыми индексами и ассоциативные массивы. Массив в PHP является фактически упорядоченным отображением. Отображение является типом, который отображает значения в ключи. Переменные массивов состоят из двух частей – индекса и элемента. Индекс массива, иногда называемый ключом массива, является значением, применяемым для идентификации или доступа к элементам массива. Индекс массива помещается в квадратные скобки. Большинство массивов используют числовые индексы, которые обычно начинаются с 0 или 1. В PHP ассоциативные массивы могут использовать строковые индексы. Оба типа массивов создаются с помощью конструкции `array()`.

### Массивы с числовыми индексами

```
$my_array = array('red', 'green', 'blue')
```

Этот код создает массив с числовым индексом с именем `$my_array`. Массиву присваивается три элемента — `red`, `green`, и `blue`. Каждый элемент идентифицируется числовым индексом.

```
$my_array[0] = 'red' // индекс 0 соответствует элементу red
$my_array[1] = 'green' // индекс 1 соответствует элементу green
$my_array[2] = 'blue' // индекс 2 соответствует элементу blue
```

Чтобы получить доступ к содержимому массива, используется имя массива и индекс. Следующий код применяется для вывода значений переменной `$my_array`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>Страница Web </title>
</head>
<body>

<p>

<?php

$my_array = array('red', 'green', 'blue');

echo "Первое значение массива — " . $my_array[0];
echo "Второе значение массива — " . $my_array[1];
echo "Третье значение массива — " . $my_array[2];

?>

</p>
</body>
</html>
Первое значение массива — red
Второе значение массива — green
Третье значение массива — blue
```

## Ассоциативные массивы

Ассоциативные массивы позволяют использовать более полезные значения индекса. Для массивов с числовыми индексами значения индекса создаются автоматически, начиная с 0. Ассоциативные массивы допускают применение числовых и строковых значений индекса. Символ между индексом и значениями (`=>`) является знаком равенства, за которым сразу следует символ больше.

```
$members = array('FName' => John, 'LName' => Smith, 'Age' => 50)
```

В этом примере члены массива содержат три элемента, однако используются строковые индексы — FName, LName и Age.

```
$members['FName'] = 'John' //индекс FName соответствует элементу John
$members['LName'] = 'Smith' // индекс LName соответствует элементу Smith
$members['Age'] = '50' // индекс Age соответствует элементу 50
```

Для доступа к содержимому массива используется имя массива и индекс. Следующий код применяется для вывода значений переменной \$members.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>Страница Web </title>
</head>
<body>

<p>

<?php

$members = array('FName' => John, 'LName' => Smith, 'Age' =>
50);

echo "The user\'s first name is " . $members['FName'];
echo "The user\'s last name is " . $members['LName'];
echo "The user\'s age is " . $members['Age'];

?>

</p>
</body>
</html>

The user's first name is John
The user's last name is Smith
The user's age is 50
```

## Функции для работы с массивами

Кроме функции `array()` система PHP включает множество других функций для работы с массивами. Следующий раздел описывает некоторые из наиболее часто используемых функций. Более обширный список доступен на Web-сайте PHP.

`count()` – функция `count` используется для подсчета числа элементов в массиве.

`sort()` – функция `sort` используется для сортировки элементов существующего массива.

`shuffle()` – функция `shuffle` используется для случайного перемешивания элементов в заданном массиве.

`sizeof()` – функция `sizeof` является синонимом (алиасом) функции `count()`.

`array_slice($array_name, offset, length)` – функция `array_slice` используется для извлечения части существующего массива. `$array_name` является именем разрезаемого массива, `offset` указывает позицию, где будет начинаться разрез, `length` указывает число элементов, которое будет вырезано из массива.

`array_merge($array_name, $array_name)` – функция `array_merge` используется для объединения или слияния двух или большего количества существующих массивов. Имена массивов разделяются запятыми.

Следующий код показывает, как применяется каждая из функций для работы с массивами.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml11-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
    <title>Страница Web </title>
</head>
<body>

<p>

<?php

//Созданы два массива

$numbers = array(50,20,18,30,10,7);
$colors = array('red', 'blue', 'green');

// определяем размер массива $numbers – 6

$array_size = sizeof($numbers);

// сортируем элементы массива $numbers – возвращает
array(7,10,18,20,30,50)
```

```

sort($numbers);

// случайным образом перемешиваем элементы массива $numbers

shuffle($numbers);

// $merged_array возвращает
array(7,10,18,20,30,50,'red','blue','green')

$merged_array = array_merge($numbers,$colors);

// вырезаем номера 18 и 20 из сортированного массива $numbers
// $slice содержит array(18,20)

$slice = array_slice($numbers, 2, 2);

?>
</p>
</body>
</html>

```

**Рисунок 3.1 – Код программы**

PHP включает также ряд predefined или глобальных массивов. Их называют также суперглобальными переменными, так как они всегда присутствуют и доступны для всех блоков сценария PHP. Ниже показаны обычно используемые суперглобальные переменные PHP.

```

$_GET[]
$_POST[]
$_REQUEST[]
$_COOKIE[]
$_FILES[]
$_SERVER[]
$_ENV[]
$_SESSION[]

```

Суперглобальные переменные PHP будут описаны в дальнейшем. Массивы имеют много применений в PHP и программировании в целом. Этот раздел представил некоторые базовые вопросы массивов PHP и описал некоторые базовые функции: это понадобится при рассмотрении более развитых свойств массивов в следующих разделах.

## **Файлы Cookies**

Файл `cookie` является сообщением от браузера Web-серверу. Браузер сохраняет сообщение в текстовом файле. Это сообщение посылается затем назад на сервер каждый раз, когда браузер запрашивает страницу с сервера.

Основное назначение `cookies` состоит в идентификации пользователей и возможной подготовке специально настроенной для них страницы Web. При посещении сайта Web, использующего `cookies`, на сайте может быть предложено заполнить форму, чтобы предоставить такую информацию, как свое имя и возможные интересы. Эта информация упаковывается в `cookie` и посылается браузеру Web, который сохраняет ее для последующего использования. Когда вы в следующий раз посещаете тот же самый сайт Web, браузер пошлет `cookie` серверу Web. Сервер может использовать эту информацию, чтобы создать индивидуализированные страницы Web. Поэтому, например, вместо обычной приветственной страницы можно увидеть приветственную страницу со своим именем.

В PHP файлы `cookies` создают с помощью функции `setcookie()`. Все данные `cookie` хранятся в глобальной переменной PHP `$_COOKIE` и доступны для последующих страниц.

`setcookie(name, value, expiration, path, domain, security)` – определяет файл `cookie`, который посылается вместе с остальными заголовками HTTP. Как и другие заголовки, файлы `cookie` должны посылаться до какого-либо вывода работы сценария (это ограничение протокола). Поэтому требуется, чтобы обращение к функции было помещено до любого вывода, включая теги и любые символы разделители. Если вывод происходит до обращения к этой функции, то `setcookie()` не выполнится и вернет `FALSE`. Если `setcookie()` выполняется успешно, то возвращается `TRUE`. Это не указывает на то, что пользователь принял `cookie`.

Параметры `setcookie()` объясняются в следующей таблице.

Параметр	Описание
<code>name</code>	Имя <code>cookie</code> . Этот идентификатор хранится в глобальной переменной <code>\$_COOKIE</code> и доступен в последующих сценариях
<code>value</code>	Значение <code>cookie</code> . Значение, связанное с идентификатором <code>cookie</code> . Хранится на компьютере пользователя, поэтому не должно содержать секретной информации
<code>expiration</code>	Время, когда истекает значение <code>cookie</code> или становится более недоступным. Это время можно задать с помощью функции <code>time()</code> . Файлы <code>cookie</code> , без заданного значения времени истечения, завершают свое существование при закрытии браузера
<code>path</code>	Указывает пути доступа на сервере, для которых <code>cookie</code> действителен или доступен. Прямая косая черта "/" говорит,

	что cookie доступен во всех папках
domain	Домен, в котором доступен cookie. Если домен не определен, по умолчанию используется хост, на котором создан cookie. Значения domain должны содержать в строке как минимум две точки ".", чтобы быть допустимыми
security	Указывает, будет ли cookie передаваться через HTTPS. Значение 1 означает, что cookie передается через защищенное соединение. Значение 0 обозначает стандартную передачу HTTP

Следующий пример демонстрирует, как cookie используется для сохранения имени пользователя посетителя. Вначале требуется ввести имя пользователя, чтобы получить доступ к ограниченному сайту. Когда имя пользователя будет создано, cookie, содержащий его, сохраняется на компьютере пользователя. Доступ в будущем возможен при извлечении cookie с компьютера пользователя.

The image shows a web form with a dark gray background. At the top, it says "New User? Create User Name". Below this is a label "User Name:" followed by a white text input box. Under the input box is a button labeled "Login". At the bottom of the form, it says "Existing User? [Enter Site](#)".

```
<?php

    if ($_REQUEST[auth] == "no")
    {
        $msg = "Вы не являетесь пользователем.
Зарегистрируйтесь.";
    }

    // Если пользователь щелкает на кнопке Login, создается
cookie,
        // содержащий его имя пользователя и IP-адрес

    if ($_POST[submit] == "Login")
    {
```

```

        $cookie_name = "user";
        $cookie_value = $_POST[uname];
        $cookie_expire = time() + 14400;

        setcookie($cookie_name,$cookie_value,$cookie_expire,"/");

        $formDisplay = "no";
    }

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<title>Страница Web </title>

<style type="text/css">

body {font:10pt arial;color:white}
div#form {background-color:gray;border:solid 1px
black;padding:10px}
input {border:solid 2px black}

</style>

<?php

    if ($formDisplay == "no")

    {

?>

<meta http-equiv='refresh'
content='0;url=siteaccess.php?auth=yes' />

<?php

    }

?>

</head>

<body>

```



```

<div id="form">
<h4 style="color:red">New User? Create User Name</h4>
<form action="setcookie.php" method="post">

<p>User Name:
<br/>
<input type="text" name="uname" size="7"/>
</p>

<input type="submit" value="Login" name="submit"/>

</form>

<h4 style="color:red">Existing User?
  <a style="color:white" href="siteaccess.php?auth=yes">Enter
Site</a></h4>

</div>
<br/>
<br/>
<?php

    echo "<span style='color:red'>" . $msg . "</span>";

?>

</body>
</html>

```

### Рисунок 3.2 - Файл setcookie.php

Построение файла siteaccess.php

```

<?php

// Если пользователь щелкает на кнопке Login,
// создается cookie, содержащий его имя пользователя и IP-адрес

if ($_REQUEST[auth] == "yes" && $_REQUEST[user])
{

    echo "Добро пожаловать " . " " . $_COOKIE[user] . " на сайт с
ограниченным доступом.
    Теперь на вашем жестком диске хранится cookie,
    и вы можете обращаться к этому сайту без регистрации при
каждом обращении";

}

else

{

```

```
        header("Location:setcookie.php?auth=no");  
    }  
  
?>
```

**Рисунок 3.2 – Файл siteaccess.php**

#### **4. Лекция: Доступ к базам данных**

PHP обеспечивает поддержку ODBC (Open DataBase Connectivity), что позволяет обращаться к любой совместимой с ODBC системе управления базами данных (СУБД), если в системе или сети доступно Имя источника данных (DSN -- Data Source Name), или доступна строка соединения без DSN

##### **Доступ ODBC**

PHP обеспечивает поддержку ODBC (Open DataBase Connectivity), что позволяет обращаться к любой совместимой с ODBC системе управления базами данных (СУБД), если в системе или сети доступно Имя источника данных (DSN — Data Source Name) или доступна строка соединения без DSN. Это включает доступ к таким реляционным базам данных, как Oracle, DB2, MS SQL Server, MySQL, и MS Access. Так как PHP включает функции доступа к базам данных MySQL без DSN, которые не требуют ODBC, в этом разделе для демонстрации методов ODBC будет применяться MS Access. Примеры можно использовать также с другими СУБД, совместимыми с ODBC.

##### **Соединения ODBC с помощью DSN**

Чтобы соединиться с базой данных, используя ODBC, сначала необходимо создать системное имя источника данных.

Вот как создается соединение ODBC с базой данных MS Access

Откройте в Панели управления значок Администрирование.

Сделайте в раскрывшемся окне двойной щелчок на значке Источники данных (ODBC).

Выберите вкладку Системный DSN.

На вкладке Системный DSN щелкните на кнопке Добавить.

Выберите Microsoft Access Driver. Щелкните на кнопке Готово.

В следующем окне щелкните на кнопке Выбрать, чтобы найти базу данных.

Задайте для базы данных Имя источника данных (DSN).

Щелкните на кнопке ОК.

Конфигурация DSN должна задаваться на компьютере, на котором размещен сайт Web. Если сайт располагается на удаленном сервере, необходимо задать конфигурацию на этом сервере.

После установления соединения ODBC можно использовать специальные функции PHP для соединения с базой данных и извлечения записей. Эти функции PHP описаны ниже.

### **Соединения ODBC без DSN**

Соединения без DSN не требуют создания DSN системного уровня для соединения с базами данных и предоставляет некоторую альтернативу DSN. Вместо использования DSN для соединения с базой данных, разработчик определяет необходимую информацию прямо в приложении. При соединении без DSN разработчик может использовать стандарты соединения, отличные от ODBC, такие, как OLE DB. Соединения без DSN должны применяться в том случае, когда отсутствует доступ к серверу для регистрации DSN.

В Microsoft Access для создания соединений без DSN используется следующая строка соединения:

```
Driver={Microsoft Access Driver (*.mdb)};  
DBQ=c:\path\to\database.mdb
```

`odbc_connect(dsn/строка соединения без dsn, имя_пользователя, пароль)` – функция, используемая для соединения с источником данных ODBC. Функция получает четыре параметра: имя источника данных (`dsn`) или строку соединения без `dsn`, имя пользователя, пароль и необязательный тип курсора. В тех случаях, когда имя пользователя, пароль, и тип курсора не требуются, параметры можно заменить пустой строкой — `' '`. `id` соединения, возвращаемый этой функцией, требуется другим функциям ODBC. Можно иметь одновременно открытыми несколько соединений, если они либо имеют различные `id`, либо используют различные имя пользователя и пароль.

`odbc_exec(id_соединения, строка_запроса SQL)` – функция, используемая для выполнения оператора SQL. Функция получает два параметра: объект соединения `id`, созданный функцией `odbc_connect()`, и оператор SQL. При возникновении ошибки возвращает `FALSE`. Возвращает множество записей, если команда SQL выполняется успешно.

`odbc_fetch_array(имя множества записей)` – используется для извлечения записей или строк из множества записей как ассоциативного

массива. Имя множества записей создается при вызове функции `odbc_exec()`. Эта функция возвращает массив строк, либо — `FALSE`.

`odbc_num_rows()` (имя множества записей) – возвращает число строк в множестве результатов ODBC. Функция возвращает -1, если возникает ошибка. Для операторов `INSERT`, `UPDATE` и `DELETE` функция `odbc_num_rows()` возвращает число затронутых строк. Для предложения `SELECT` это может быть число доступных строк. Примечание: использование `odbc_num_rows()` для определения числа доступных строк после оператора `SELECT` возвращает -1 для драйверов MS Access.

`odbc_close(id соединения)` – закрывает соединение с сервером базы данных, связанное с данным идентификатором соединения.

## Добавление записей

Применяя функции ODBC, рассмотренные в предыдущем разделе, вместе с языком SQL можно добавлять записи в таблицы базы данных. Записи добавляются в таблицу базы данных с помощью формы, представляющей поля ввода данных. Затем с помощью кнопки вызывается сценарий PHP для записи новой информации в таблицу с помощью команды SQL `INSERT`. Типичная форма ввода для добавления новой записи в таблицу Survey показана ниже.

**VisitorSurvey.php**

<b>Name</b> <input type="text"/>	<b>Email</b> <input type="text"/>
<b>Web Connection</b> <input type="text" value="Dial Up"/>	<b>Residence (City/ST/Country)</b> <input type="text"/>
<b>Age</b> <input type="radio"/> Under 20 <input type="radio"/> 21-25 <input type="radio"/> 26-30 <input type="radio"/> Over 30	<b>Gender</b> <input type="radio"/> Male <input type="radio"/> Female
<b>Comments</b> <div><input type="text"/></div>	
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

Поля формы именуются соответствующим образом:

```
Name - 'Name', Email - 'Email',  
Web Connection - 'Connection',  
Residence (City/ST/Country) - 'Residence',  
Age - 'Age', Gender - 'Gender',  
Comments - 'Comments'.
```

Прежде чем переходить к коду, покажем синтаксис оператора SQL  
INSERT:

```
INSERT INTO TableName (FieldName1 [,FieldName2]...) VALUES  
(Value1 [,Value2]...)
```

Более подробно оператор INSERT рассматривается в других разделах.

Следующий код применяется для обработки данных формы

VisitorSurvey.php:

VisitorSurvey.php

```
<?php
```

```
if ($_POST[submit] == "Submit")  
{
```

```
// Получение данных формы и присвоение скалярным переменным
```

```
$Name = $_POST[Name];  
$Email = $_POST[Email];  
$Connection = $_POST[Connection];  
$Residence = $_POST[Residence];  
$Age = $_POST[Age];  
$Gender = $_POST[Gender];  
$Comments = $_POST[Comments];
```

```
//Установка соединения с базой данных
```

```
$conn = odbc_connect('Driver={Microsoft Access Driver (*.mdb)};  
DBQ=c:\path\to\database.mdb',' ','');
```

```
// Оператор SQL
```

```
$sql = "INSERT INTO Survey " .  
      "(Name,Email,Connection,Residence,Age,Gender,Comments)  
VALUES ('$Name', '$Email', '$Connection',  
      '$Residence', '$Age', '$Gender', '$Comments')";
```

```
//Выполнение оператора SQL и сохранение результатов в множестве  
записей
```

```
$rs = odbc_exec($conn,$sql);
```

```
odbc_close($conn);  
  
}  
  
?>
```

Рисунок 4.1 – Обработка данных формы

После щелчка на кнопке отправки создается суперглобальный массив `$_POST`, содержащий значения формы. Значения массива присваиваются скалярным переменным. Это упрощает кодирование оператора SQL. Затем выполняется оператор `odbc_connect()`. Этому оператору требуется три параметра – DSN или строка соединения без DSN, имя пользователя и пароль. Здесь для соединения с базой данных Access используется строка соединения без DSN. Так как Access для доступа к данным не требует имя пользователя или пароль, то параметры `username` и `password` кодируются как значения `NULL`. Ссылка на соединение сохраняется в переменной `$conn`. Это пример ссылочной переменной PHP.

В отличие от скаляров или переменных массивов, ссылочные переменные не применяются в программе непосредственно, но часто используются в качестве параметров для других функций. После оператора соединения составляется оператор SQL `INSERT` и присваивается переменной `$sql`. Затем функция `odbc_exec()` выполняет оператор SQL, создающий множество записей (множество записей из базы данных). Это множество записей присваивается `$rs`, другой ссылочной переменной PHP. Отметим, что функция `odbc_exec()` требует два параметра — `$conn` (ссылка на текущее соединение базой данных) и `$sql` (ссылка на текущий оператор SQL). В конце вызывается функция `odbc_close()` для закрытия текущего соединения с базой данных.

Для приложений такого типа обычно желательно выполнить проверку данных, прежде чем добавлять данные в таблицу базы данных. Это необходимо сделать перед тем, как создавать соединение с базой данных, используя технику, рассмотренную в [лекции 7](#).

В случае ошибки кодирования данные не заносятся в таблицу базы данных, а PHP выводит предупреждение или сообщение об ошибке. В ситуациях такого типа полезно подавить эти сообщения, добавить код для проверки ошибок вручную и создать более понятный для пользователя вывод. Это можно сделать сразу после оператора `odbc_exec()`, проверяя статус вновь созданного множества записей — `$rs`.

Предположим в предыдущем коде, что функция `odbc_exec()` содержит вместо `$sql` параметр `$sqlString`. PHP немедленно остановит выполнение страницы и выведет следующее сообщение:

```
Warning: odbc_exec() [function.odbc-exec]:  
SQL error: [Microsoft][ODBC Microsoft Access Driver]  
    Invalid use of null pointer ,  
SQL state S1009 in SQLExecDirect in  
    C:\ApacheRoot\VisitorSurvey.php on line 38
```

Ошибку PHP можно подавить с помощью оператора управления ошибками "@". После подавления ошибки можно добавить код для создания более понятного пользователю ответа. Такой подход показан ниже:

Файл VisitorSurvey.php

```
<?php  
  
if ($_POST[submit] == "Submit")  
{  
  
    //Извлечение данных формы и присвоение скалярным переменным  
  
    $Name = $_POST[Name];  
    $Email = $_POST[Email];  
    $Connection = $_POST[Connection];  
    $Residence = $_POST[Residence];  
    $Age = $_POST[Age];  
    $Gender = $_POST[Gender];  
    $Comments = $_POST[Comments];  
  
    //Установление соединения с базой данных  
  
    $conn = @odbc_connect('Driver={Microsoft Access Driver (*.mdb)};  
    DBQ=c:\path\to\database.mdb','','');  
  
    //Оператор SQL  
  
    $sql = "INSERT INTO Survey " .  
        " (Name,Email,Connection,Residence,Age,Gender,Comments)  
    VALUES ('$Name', '$Email', '$Connection',  
        '$Residence', '$Age', '$Gender', '$Comments')";  
  
    //Выполнение оператора SQL и сохранение результатов в множестве  
    записей  
  
    $rs = @odbc_exec($conn,$sqlstring);  
  
    if (!$rs)  
    {  
  
        echo "Произошла ошибка. Попробуйте еще раз. ";  
  
    }  
  
    else
```

```

{
echo "Запись была успешно добавлена.";
}

odbc_close($conn);
}

?>

```

Рисунок 4.2 – Обработка ошибок

После функции `odbc_exec()` используется оператор `if` для проверки статуса множества записей — `$rs`. Если множество записей успешно создано, то выводится сообщение "Запись была успешно добавлена". Если возникает проблема, выводится сообщение "Произошла ошибка. Попробуйте еще раз."

## Выбор записей

Кроме использования функций ODBC с оператором SQL `INSERT` для добавления записей в базу данных, можно также извлекать записи из таблицы базы данных с помощью оператора SQL `SELECT`. Типичная форма ввода для выбора существующих записей из таблицы `Directory` показана ниже. В этом примере фиктивная компания `Company XYZ`, имеет онлайн-форму, которая позволяет пользователям ввести фамилию сотрудника и найти имя и фамилию сотрудника, номер телефона и адрес e-mail.

Company XYZ Directory	
<input type="text"/>	<input type="button" value="Search"/>

Оператор SQL `SELECT` показан ниже:

```

SELECT * | [DISTINCT] field1 [,field2]... FROM TableName WHERE
criteria
ORDER BY FieldName1 [ASC|DESC] [,FieldName2 [ASC|DESC] ]...

```

Более подробное рассмотрение оператора `SELECT` дано в [разделе 10](#).

Следующий код используется для обработки формы

`DirectorySearch.php`:

`DirectorySearch.php`



```

<?php

if ($_POST[submit] == "Search")
{

    //Извлечение данных формы

    $string = $_POST['search'];

    //Установление соединения с данными

    $conn = odbc_connect('Driver={Microsoft Access Driver
(*.mdb)}; DBQ=c:\path\to\database.mdb',' ',' ');

    //Выполнение оператора SQL SELECT

    $sql = "SELECT * FROM Directory WHERE LName = '$string'";

    $rs = odbc_exec($conn, $sql);
    }

?>

<!DOCTYPE html PUBLIC "-//W3C/DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml11-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
<title>Страница Web</title>

<style>

    body {margin:15px;font:10pt Verdana}
    td {vertical-align:top;border:solid 1px gray}
    input,textarea{border:0px}

    </style>

</head>

<body>

    <form action="DirectorySearch.php" method="post">
    <p> Введите ниже фамилию и щелкните на кнопке "Search",
чтобы найти номер телефона сотрудника и адрес e-mail </p>
    <table>
    <tr>
    <td colspan="2">Company XYZ Directory</td>
    </tr>
    <td><input type="text" size="15" name="SearchName"/></td>

```

```

<td><input type="submit" value="Search" name="submit"/>
</tr>
</table>
</form>

<div>

<?php

if(!empty($_POST))
{
    while($row = odbc_fetch_array($rs))
    {
        echo "Name: " . $row['FName'] . " ";
        echo $row['LName'] . "<br/>";
        echo "Telephone: " . $row['Telephone'] . "<br/>";
        echo "Email: " . $row['Email'] . "<br/>";
    }

    odbc_close($conn);
}
?>

</div>
</body>
</html>

```

Рисунок 4.3 – Обработка формы

Эта страница содержит два блока кода PHP. Первый выполняется, когда делается щелчок на кнопке отправки "Search". Введенное пользователем имя (*first name*) присваивается скалярной переменной '\$string'. Затем с базой данных устанавливается соединение и выполняется оператор SQL `SELECT` для выбора всех (\*) полей записей таблицы, где значение поля 'lastname' совпадает со строкой фамилии, введенной пользователем. Наконец, выполняется оператор SQL. Если найдены подходящие записи, то множество записей присваивается переменной '\$rs'.

Второй блок кода появляется в разделе тела XHTML-документа. Код, содержащий операторы `echo` или `print`, помещается обычно между открывающим и закрывающим тегами `<body>`, чтобы он выводился или форматировался в соответствии с другими элементами страницы. Операторы `echo` и `print` в блоках PHP, появляющиеся выше `<html>`, всегда появляются вверху страницы и предшествуют всем другим ее элементам.

Назначение этого блока кода состоит в выводе записей, извлеченных при выполнении оператора SQL в предыдущем блоке кода. Сначала оператор `if` используется для определения, что массив `$_POST` не

является пустым. Если массив пустой, то это означает, что форма не была отправлена и никакие записи не были извлечены. Если этот условный оператор будет отсутствовать, то возникнет ошибка, так как массив `odbc_fetch_array` не будет содержать никаких значений.

Затем используется цикл `while` для итераций на множестве записей. Во время каждой итерации по множеству записей, функция `odbc_fetch_array()` создает ассоциативный массив (здесь массив называется `$row`), содержащий значения полей текущей записи. Индексы массива соответствуют именам полей формы, а элементы массива — значениям полей. Каждая запись затем выводит `$row['FName']` — значение поля 'FirstName', `$row['LName']` — значение поля 'LastName', `$row['Telephone']` — значение поля 'Telephone', и `$row['Email']` — значение поля 'Email'. Этот процесс продолжается, при этом каждый раз массив `$row` содержит новые значения, пока не будет достигнут конец множества записей.

После вывода всех записей соединение с базой данных закрывается с помощью функции `odbc_close()`.

Ниже показан пример вывода, созданного после поиска в каталоге компании XYZ.

Company XYZ Directory

Name: Molly Douglas  
Telephone: 5553  
Email: mdouglas@php.net

Name: John Douglas  
Telephone: 8883  
Email: jdouglas@php.net

Если пользователь ищет фамилию, которая не существует в таблице базы данных, то никакие записи не выводятся. Чтобы избежать путаницы, приведенный выше сценарий можно немного изменить, выводя сообщение, если не будет найдено подходящих записей.

DirectorySearch.php

```
<?php

if (!empty($_POST))
{
```

```

$results = 0;
while($row = odbc_fetch_array($rs))

{
    echo "Name: " . $row['FName'] . " ";
    echo $row['LName'] . "<br/>";
    echo "Telephone: " . $row['Telephone'] . "<br/>";
    echo "Email: " . $row['Email'] . "<br/>";

    $results += 1;
}

if ($results == 0)

{

    echo "Ничего не найдено!";

}

odbc_close($conn);
}
?>

</div>
</body>
</html>

```

Модифицированный сценарий, показанный выше, добавляет счетчик записей `$results` для подсчета числа выводимых записей. В начале значение счетчика задается равным 0. Если множество записей существует, то цикл `while` будет выполнять итерации на каждой записи и увеличивать значение счетчика на 1. После окончания цикла проверяется значение счетчика. Если это значение равно 0, то никаких записей не было найдено, и выводится сообщение "Ничего не найдено!"

## Удаление записей

Оператор SQL `DELETE` используется для удаления существующих в базе данных записей.

Синтаксис оператора SQL `DELETE` показан ниже:

```
DELETE FROM Имя_таблицы WHERE критерий
```

Более подробно оператор `DELETE` рассматривается в [разделе 10](#).

Следующая форма представляет запись пользователя, которая будет удалена из таблицы базы данных `Personnel` (Персонал). Щелчок на

кнопке Delete вызывает процедуру PHP, которая выполняет оператор SQL DELETE, чтобы удалить эту запись из таблицы базы данных.



First Name:	Molly
Last Name:	Douglas
Telephone:	5555
Email:	mdouglas@php.net

Delete Record

Кроме показанных выше элементов управления формы страница содержит также скрытое текстовое поле с именем "AutoNum" со значением, равным полю AutoNum таблицы базы данных. Это поле используется для уникальной идентификации каждой записи. Следующий код демонстрирует работу страницы:

DirectorySearch.php

```
<?php
```

```
if ($_POST['submitb']=="Delete Record")
{
    $conn = odbc_connect('Driver={Microsoft Access Driver
(*.mdb)}; DBQ=c:\path\to\database.mdb','', '');
    $sqlDelete = "DELETE FROM Personnel WHERE AutoNum =" .
$_POST['AutoNum'];
    $rsDelete = odbc_exec($conn,$sqlDelete);

    if(odbc_num_rows($rsDelete) == 1)
    {
        echo "Запись успешно удалена!";
    }

    odbc_close($conn);
}

?>
```

Рисунок 4.4 – Удаление записей

После щелчка на кнопке "Delete Record" устанавливается соединение с базой данных Access. Затем формируется оператор SQL DELETE для

удаления записи из таблицы `Personnel` со значением поля `AutoNum` равным значению скрытого текстового поля `AutoNum`. Затем выполняется оператор SQL. Результаты функции `odbc_exec()` присваиваются переменной `$rsDelete`. Последний шаг состоит в проверке, что удаление записи было успешным, и в выводе подтверждающего сообщения. Функция `odbc_num_rows()` применяется для определения числа строк в результатах ODBC или числа строк, затронутых оператором `odbc_exec()`. Так как удаляется одна запись, то запись будет удалена успешно, если результат функции `odbc_num_rows()` будет равен 1. В конце соединение с базой данных закрывается.

## Изменение записей

Оператор SQL `UPDATE` используется для изменения или обновления существующих записей базы данных.

Синтаксис оператора SQL `UPDATE` показан ниже:

```
UPDATE ИмяТаблицы SET (FieldName1=value1
[,FieldName2=value2]...)
WHERE критерий
```

Следующая форма представляет запись пользователя, которая будет обновлена в таблице базы данных `Personnel`. Модификация любых полей, связанных с записью, и щелчок на кнопке `Update Record` вызывает процедуру PHP, которая выполняет оператор SQL `UPDATE`, чтобы обновить эту запись в таблице базы данных.

First Name:	<input type="text" value="Scott"/>
Last Name:	<input type="text" value="Jones"/>
Telephone:	<input type="text" value="3333"/>
Email:	<input type="text" value="sjones@php.net"/>
<input type="button" value="Update Record"/>	

Кроме показанных выше элементов управления формы страница содержит также скрытое текстовое поле с именем `"AutoNum"` со значением, равным значению поля `AutoNum` в таблицы базы данных. Это поле используется для уникальной идентификации каждой записи. Следующий код демонстрирует работу страницы:

`DirectorySearch.php`

`<?php`

```

if ($_POST['submitb']=="Update Record")
{

    $new_fname = $_POST['FName'];
    $new_lname= $_POST['LName'];
    $new_telephone = $_POST['Telephone'];
    $new_email = $_POST['Email'];

    $conn = odbc_connect('Driver={Microsoft Access Driver
(*.mdb)}; DBQ=c:\path\to\database.mdb',' ',' ');
    $sqlUpdate = "UPDATE Personnel SET (FName = '$new_fname',
LName = '$new_lname', Telephone = '$new_telephone',
Email = '$new_email') WHERE AutoNum =" . $_POST['AutoNum'];
    $rsUpdate = odbc_exec($conn,$sqlUpdate);

    if(odbc_num_rows($rsUpdate) == 1)
    {

        echo "Запись успешно обновлена!";

    }

    odbc_close($conn);

}

?>

```

**Рисунок 4.5 – Обновление записей**

После щелчка на кнопке "Update Record" текущие значения полей формы присваиваются скалярным переменным. Этот шаг не является обязательным, однако он упрощает кодирование оператора SQL UPDATE. Затем устанавливается соединение с базой данных Access. После соединения с базой данных формируется оператор SQL UPDATE для обновления записи в таблице Personnel со значением поля AutoNum, равным значению скрытого текстового поля AutoNum. Затем выполняется оператор SQL. Результаты функции odbc\_exec() присваиваются переменной \$rsUpdate. Последний шаг состоит в проверке, что обновление записи было успешно, и в выводе подтверждающего сообщения.

Функция odbc\_num\_rows() используется для определения числа строк в результате ODBC или числа строк, затронутых оператором odbc\_exec(). Так как обновляется только одна запись, то обновление записи проходит успешно, если результат функции odbc\_num\_rows() будет равен 1. В конце соединение с базой данных закрывается.

## 5. Лекция: Доступ к базе данных MySQL

Система PHP предоставляет поддержку для MySQL с помощью набора функций, которые можно использовать для манипуляции данными MySQL. База данных MySQL стала самой популярной в мире базой данных с открытым исходным кодом в связи с ее высокой производительностью, надежностью и легкостью использования

### Доступ к MySQL

База данных MySQL® стала самой популярной в мире базой данных с открытым исходным кодом — благодаря ее высокой производительности, надежности и легкости использования. Существует более 6 миллионов установок этой базы данных, начиная от больших корпораций и до специализированных встроенных приложений.



Система PHP предоставляет поддержку для MySQL с помощью набора функций, которые можно использовать для манипуляции данными MySQL. Задача этого учебника состоит в том, чтобы познакомить с этими функциями, используемыми обычно в ориентированных на данные приложениях PHP для извлечения, обновления, добавления и удаления данных.

Используемые обычно функции PHP для MySQL описаны ниже.

`mysql_connect(имя сервера MySQL, имя пользователя, пароль)` – открывает соединение с сервером MySQL.

`mysql_select_db(имя базы данных, идентификатор_соединения)` – выбирает базу данных, расположенную на сервере MySQL. Параметр "имя базы данных" относится к активной базе данных на сервере MySQL, который был открыт с помощью функции `mysql_connect`. "Идентификатор\_соединения" является ссылкой на текущее соединение с MySQL.

`mysql_query(запрос sql)` – посылает запрос активной в данный момент базе данных.

`mysql_fetch_array(resource result)` – возвращает массив, который соответствует извлеченной строке, и перемещает внутренний указатель данных вперед.



`mysql_affected_rows(resource result)` – определяет число строк, затронутых предыдущей операцией SQL.

`mysql_close(link_identifier)` – закрывает соединение MySQL.

Прежде чем можно будет применять эти функции для создания приложений обработки данных с помощью MySQL, необходимо получить подходящий доступ к серверу MySQL. Для этого требуется учетная запись пользователя и пароль с полномочиями доступа к базе данных и таблицам, содержащим данные, а также имя хоста сервера MySQL или IP-адрес.

При работе с сервером MySQL полезно также использовать инструменты управления с графическим интерфейсом, которые обеспечивают более легкий интерфейс использования данных. Популярными инструментами являются: SQLyog (доступный на <http://www.webyog.com>) и MySQL Administrator (доступный на <http://www.mysql.com/products/tools/>).

## **Добавление записей**

С помощью рассмотренных в предыдущем разделе функций MySQL и языка SQL можно добавлять записи в таблицу базы данных. Записи добавляются с помощью формы, предоставляющей области для ввода информации. Нажатие кнопки вызывает затем сценарий PHP для записи новой информации в таблицу с помощью команды SQL `INSERT`.

Типичная форма ввода для добавления новой записи в таблицу `Survey` показана ниже.

VisitorSurvey.php

<b>Name</b>	<b>Email</b>
<b>Web Connection</b> Dial Up	<b>Residence (City/ST/ Country)</b>
<b>Age</b> <input type="radio"/> Under 20 <input type="radio"/> 21-25 <input type="radio"/> 26-30 <input type="radio"/> Over 30	<b>Gender</b> <input type="radio"/> Male <input type="radio"/> Female
<b>Comments</b>	
<div>Submit</div> <div>Reset</div>	

Поля формы именованы соответствующим образом:

```
Name - 'Name', Email - 'Email', Web Connection - 'Connection',
Residence (City/ST/Country) - 'Residence', Age - 'Age', Gender -
'Gender',
Comments - 'Comments'.
```

Прежде чем подробно рассматривать код, будет полезно посмотреть на синтаксис оператора SQL `INSERT`:

```
INSERT INTO TableName (FieldName1 [,FieldName2]...) VALUES
(Value1 [,Value2]...)
```

Более подробное рассмотрение оператора `INSERT` дано в приложении.

Следующий код используется для обработки данных формы

VisitorSurvey.php:

VisitorSurvey.php

```
<?php
```

```
if ($_POST[submit] == "Submit")
{
```

```

//Получение данных формы и присвоение скалярным переменным

$Name = $_POST[Name];
$Email = $_POST[Email];
$Connection = $_POST[Connection];
$Residence = $_POST[Residence];
$Age = $_POST[Age];
$Gender = $_POST[Gender];
$Comments = $_POST[Comments];

//Установление соединения с базой данных

$conn = mysql_connect('localhost','root','хыхыху');

//Выбор базы данных MySQL

$db = mysql_select_db('Membership', $conn);

//Оператор SQL

$sql = "INSERT INTO Survey " .
      "(Name,Email,Connection,Residence,Age,Gender,Comments)
VALUES ('$Name', '$Email', '$Connection',
        '$Residence', '$Age', '$Gender', '$Comments')";

//Выполнение оператора SQL и сохранение результатов в множестве
записей

$rs = mysql_query($sql,$conn);

mysql_close($conn);

}

?>

```

Рисунок 5.1 – Обработки данных формы

После нажатия кнопки отправки Submit создается суперглобальный массив `$_POST`, содержащий значения из формы. Значения массива присваиваются скалярным переменным. Это упрощает кодирование оператора SQL. Затем выполняется оператор `mysql_connect()`. Этот оператор требует три параметра – имя хоста сервера MySQL, имя пользователя, и пароль. Здесь для соединения с базой данных MySQL используется строка соединения без DSN. Ссылка на соединение хранится в переменной `$conn`. Это пример ссылочной переменной PHP. В отличие от скалярных переменных и массивов, ссылочные переменные не применяются непосредственно в программе, но часто используются как параметры для других функций. После соединения с сервером базы данных MySQL следующий шаг состоит в выборе базы данных. Так как экземпляр сервера MySQL может содержать большое число баз данных, то задействуется функция `mysql_select_db` для выбора одной из них для

использования в приложении. Эта функция требует два параметра – имя базы данных и ссылку на соединение MySQL. Вслед за выбором базы данных формируется оператор SQL `INSERT` и присваивается переменной `$sql`. Затем функция `mysql_query()` выполняет оператор SQL, создающий множество записей (множество записей базы данных). Это множество записей присваивается переменной `$rs`, еще одной ссылочной переменной PHP. Отметим, что функция `mysql_query()` требует два параметра — `$sql` (ссылка на текущий оператор SQL) и `$conn` (ссылка на текущее соединение с базой данных). Наконец, вызывается функция `close()` для закрытия текущего соединения с базой данных. Для таких приложений обычно желательно выполнять проверку данных, прежде чем заносить данные в таблицу базы данных. Это необходимо делать перед установлением соединения с базой данных с помощью методов, рассмотренных в [лекции 7](#).

В случае ошибки кодирования данные не заносятся в таблицу базы данных, а PHP выведет предупреждение или сообщение об ошибке. В такой ситуации полезно подавить эти критические сообщения, добавить код для проверки ошибок вручную и сгенерировать более понятный для пользователя вывод. Это можно делать сразу после оператора `mysql_query()`, проверяя статус вновь созданного множества записей — `$rs`.

Предположим, что в предыдущем коде функция `mysql_query()` содержит вместо `$sql` параметр `$sqlString`. В этом случае PHP немедленно прекратит выполнение страницы и выведет сообщение об ошибке.

Вывод сообщения об ошибке PHP можно подавить с помощью оператора управления ошибками `@`. При подавленном сообщении об ошибке можно добавить код для создания более понятного пользователю сообщения. Такой подход показан ниже:

VisitorSurvey.php

```
<?php
```

```
if ($_POST[submit] == "Submit")
{
```

```
//Получение данных формы и присвоение скалярным переменным
```

```
$Name = $_POST[Name];
$Email = $_POST[Email];
$Connection = $_POST[Connection];
$Residence = $_POST[Residence];
$Age = $_POST[Age];
$Gender = $_POST[Gender];
$Comments = $_POST[Comments];
```

```
//Установление соединения с базой данных
```

```

$conn = @mysql_connect('localhost','root','xyxyxy');

//Выбор базы данных MySQL

$db = @mysql_select_db('Membership', $conn);

//Оператор SQL

$sql = "INSERT INTO Survey " .
      " (Name,Email,Connection,Residence,Age,Gender,Comments)
VALUES ('$Name', '$Email', '$Connection',
        '$Residence', '$Age', '$Gender', '$Comments')";

//Выполнение оператора SQL и сохранение результатов в виде
множества записей

$rs = @mysql_query($sqlstring,$conn);

if (!$rs)
{

echo "Произошла ошибка. Попробуйте еще раз.";

}

else

{

echo "Запись была успешно добавлена.";

}

mysql_close($conn);

}

?>

```

Рисунок 5.2 - Обработка ошибок

Вслед за функцией `mysql_query()` используется оператор `if` для проверки статуса множества записей `$rs`. Если множество записей успешно создается, выводится сообщение "Запись была успешно добавлена". Если возникает проблема, выводится сообщение "Произошла ошибка. Попробуйте еще раз."

## Выбор записей

Кроме применения функций MySQL с оператором SQL `INSERT` для добавления записей в базу данных, можно также извлекать записи из таблицы базы данных с помощью оператора SQL `SELECT`.

Типичная форма ввода для выбора существующих записей из таблицы Directory показана ниже. В этом примере фиктивная компания, Company XYZ, имеет онлайн форму, которая позволяет пользователям ввести фамилию сотрудника и найти полное имя сотрудника, номер телефона и адрес e-mail.

Company XYZ Directory	
<input type="text"/>	<input type="button" value="Search"/>

Оператор SQL SELECT показан ниже:

```
SELECT * | [DISTINCT] field1 [,field2]... FROM TableName WHERE  
criteria  
ORDER BY FieldName1 [ASC|DESC] [,FieldName2 [ASC|DESC] ]...
```

Более подробно оператор SELECT рассматривается в приложении.

Следующий код используется для обработки формы  
DirectorySearch.php:

DirectorySearch.php

```
<?php
```

```
if ($_POST[submit] == "Search")  
{
```

```
    //Получение данных формы
```

```
    $string = $_POST['search'];
```

```
    //Установление соединения с данными
```

```
    $conn = mysql_connect('localhost','root','хухуху');
```

```
    //Выбор базы данных
```

```
    $db = mysql_select_db('Membership', $conn);
```

```
    //Создание оператора SQL SELECT
```

```
    $sql = "SELECT * FROM Directory WHERE LName = '$string'";
```

```
    $rs = mysql_query($sql, $conn);  
}
```

```
?>
```

```
<!DOCTYPE html PUBLIC "-//W3C/DTD/XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml11-
transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
```

```
<head>
```

```
<title>Страница Web </title>
```

```
<style>
```

```
    body {margin:15px;font:10pt Verdana}
    td {vertical-align:top;border:solid 1px gray}
    input,textarea{border:0px}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<form action="DirectorySearch.php" method="post">
<p>Введите ниже фамилию и нажмите кнопку "Search",
чтобы найти номер телефона и адрес e-mail сотрудника</p>
<table>
<tr>
<td colspan="2">Company XYZ Directory</td>
</tr>
<td><input type="text" size="15" name="SearchName"/></td>
<td><input type="submit" value="Search" name="submit"/>
</td>
</tr>
</table>
</form>
```

```
<div>
```

```
<?php
```

```
if(!empty($_POST))
{
    while($row = mysql_fetch_array($rs))
    {
        echo "Name: " . $row['FName'] . " ";
        echo $row['LName'] . "<br/>";
        echo "Telephone: " . $row['Telephone'] . "<br/>";
        echo "Email: " . $row['Email'] . "<br/>";
    }
    mysql_close($conn);
}
?>
```

```
</div>
</body>
</html>
```

Рисунок 5.3 – Пример обработки

Эта страница содержит два блока кода PHP. Первый выполняется, когда нажимается кнопка отправки формы "Search". Введенная пользователем фамилия присваивается скалярной переменной '\$string'. Затем создается соединение с базой данных и формируется оператор SQL `SELECT` для выбора всех (\*) полей таблицы записей, в которых поле 'lastname' совпадает со строкой фамилии, введенной пользователем. В конце выполняется оператор SQL. Если найдены подходящие записи, то множество записей присваивается переменной '\$rs'.

Второй блок кода появляется в теле документа XHTML. Код, содержащий операторы `echo` или `print`, помещается обычно между открывающим и закрывающим тегами `<body>`, так что он может выводиться или форматироваться в соответствии с другими элементами страницы. Операторы `echo` и `print`, появляющиеся в блоках PHP, закодированные выше тега `<html>`, всегда появляются в верху страницы и предшествуют всем другим ее элементам.

Назначение этого блока кода состоит в выводе извлеченных записей, если в предыдущем блоке кода был выполнен оператор SQL. Сначала используется оператор `if` для проверки, что массив `$POST` не является пустым. Если этот массив будет пустым, то это означает, что форма не была отправлена, и никакие записи не были извлечены. Если этот условный оператор отсутствует, то будет возникать ошибка, так как массив `mysql_fetch_array` не будет содержать никаких значений.

Затем используется цикл `while` для итераций по множеству записей. Во время каждой итерации по множеству записей функция `mysql_fetch_array()` создает ассоциативный массив (здесь этот массив назван `$row`), содержащий значения полей текущей записи. Индексы массива соответствуют именам полей формы, а элемент массива соответствует значению поля. Каждая запись затем выводит `$row['FName']` — значение поля 'FirstName', `$row['LName']` -- значение поля 'LastName', `$row['Telephone']` – значение поля 'Telephone', и `$row['Email']` – значение поля 'Email'. Этот процесс продолжается, пока не будет достигнут конец множества записей, при этом массив `$row` каждый раз будет содержать новые значения.

После вывода всех записей соединение с базой данных закрывается с помощью функции `mysql_close()`.

Ниже представлен пример вывода, созданного после поиска в каталоге.



Company XYZ Directory

Name: Molly Douglas

Telephone: 5553

Email: mdouglas@php.net

Name: John Douglas

Telephone: 8883

Email: jdouglas@php.net

Если пользователь ищет фамилию, которая не существует в таблице базы данных, то никакие записи не выводятся. Чтобы предотвратить путаницу, приведенный выше сценарий можно немного изменить так, что будет выводиться соответствующее сообщение, если подходящие записи не будут найдены.

DirectorySearch.php

```
<?php

if(!empty($_POST))
{

    while($row = mysql_fetch_array($rs))
    {

        echo "Name: " . $row['FName'] . " ";
        echo $row['LName'] . "<br/>";
        echo "Telephone: " . $row['Telephone'] . "<br/>";
        echo "Email: " . $row['Email'] . "<br/>";

    }

    if (mysql_affected_rows($rs) == 0)
    {

        echo "No records found!";

    }

    mysql_close($conn);
}
?>
```

```
</div>
</body>
</html>
```

Показанный выше измененный сценарий содержит дополнительно функцию `mysql_affected_rows()`. Эта функция требует один параметр – ссылку на текущее множество записей `$rs` и определяет число строк, затронутых последней операцией SQL. Если возвращаемое функцией `mysql_affected_rows()` значение равно 0, то оператор SQL `SELECT` не затронул ни одной строки. Поэтому подходящих записей найдено не было.

## Удаление записей


Оператор SQL `DELETE` используется для удаления существующих записей в базе данных.

Синтаксис оператора SQL `DELETE` показан ниже:

```
DELETE FROM Имя_Таблицы WHERE критерий
```

Более подробно оператор `DELETE` рассматривается в приложении.

Следующая форма представляет запись пользователя, которая будет удалена из таблицы базы данных `Personnel`. Щелчок на кнопке `Delete` вызывает процедуру PHP, которая выполняет оператор SQL `DELETE` для удаления этой записи из таблицы базы данных.



First Name:	Molly
Last Name:	Douglas
Telephone:	5555
Email:	mdouglas@php.net

Delete Record

Кроме показанных выше элементов управления формы, страница включает также скрытое текстовое поле с именем "AutoNum" со значением, равным полю `AutoNum` таблицы базы данных. Это поле используется для уникальной идентификации каждой записи. Следующий код демонстрирует, как работает страница:

```
DirectorySearch.php
```

```
<?php
```

```

if ($_POST['submitb']=="Delete Record")
{

    $conn = mysql_connect('localhost','root','xyxyxy');
    $db = mysql_select_db('Membership',$conn);
    $sqlDelete = "DELETE FROM Personnel WHERE AutoNum =" .
$_POST['AutoNum'];
    $rsDelete = mysql_query($sqlDelete,$conn);

    if(mysql_affected_rows($rsDelete) == 1)
    {

        echo "Запись успешно удалена!";

    }

    mysql_close($conn);

}

?>

```

После нажатия кнопки "Delete Record" устанавливается соединение с базой данных MySQL. Затем создается оператор SQL DELETE для удаления записи из таблицы Personnel со значением поля AutoNum, равным значению скрытого текстового поля AutoNum. Затем оператор SQL выполняется. Результаты работы функции mysql\_query() присваиваются переменной \$rsDelete. Последний шаг состоит в проверке, что удаление записи прошло успешно, и в выводе подтверждающего сообщения. Функция mysql\_affected\_rows() используется для определения числа строк в множестве результатов ODBC или числа строк, затронутых оператором mysql\_query(). Так как будет удалена только одна запись, то результат mysql\_affected\_rows() равный 1 означает, что запись удалена успешно. В конце соединение с базой данных закрывается.

## 6. Лекция: Доступ к файлам и папкам

PHP предоставляет доступ к файлам в операционных системах Windows и Unix для чтения, записи или добавления содержимого. Эта лекция описывает, как использовать PHP для открытия файлов в системах Windows

### Открытие файлов

PHP предоставляет доступ к файлам в операционных системах Windows и Unix для чтения, записи или добавления содержимого. Этот раздел описывает, как использовать PHP для открытия файлов в системах Windows.

PHP содержит функции `fopen()` и `fclose()` для работы с файлами. Обе функции определяются ниже.

`fopen(имя_файла, режим)` - функция используется для открытия файла. Для функции требуется задать имя файла и режим работы. Она возвращает указатель на файл, который содержит информацию о файле и используется в качестве ссылки.

`fclose(указатель_ресурса)` - функция используется для закрытия файла. Для функции требуется указатель файла, созданный при открытии файла с помощью функции `fopen()`. Возвращает `TRUE` при успешной работе или `FALSE` при отказе.

Имя файла является полным путем доступа к файлу, который требуется создать или открыть. Этот путь доступа может быть относительным путем доступа к файлу: `"/Documents and Settings/Administrator/PHP/myfile.txt"` или абсолютным путем доступа к файлу: `"E:/MyFiles/PHP/myfile.txt"`. Для каждого определенного каталога необходимо иметь подходящие права NTFS для создания, модификации или удаления файлов.

Режим может быть одним из следующих.

Режимы, используемые в `fopen()`

Режим	Применение
r	Открывает существующий файл с целью чтения из него данных. Указатель файла помещается в начале файла
r+	Открывает существующий файл с целью чтения или записи данных. Указатель файла помещается в начале файла
w	Открывает файл для записи. Если файл не существует, то он создается. Если файл существует, то указатель файла помещается в начале файла и функция удаляет все существующее содержимое файла
w+	Открывает файл для чтения и записи. Если файл не существует, то он создается. Если файл существует, то указатель файла помещается в начале файла и функция удаляет все существующее содержимое файла
a	Открывает файл для записи. Если файл не существует, то он создается. Если файл существует, то указатель файла помещается в конце файла

a+	Открывает файл для чтения и записи. Если файл не существует, то он создается. Если файл существует, то указатель файла помещается в конце файла
----	---

Следующий пример показывает, как открыть файл для чтения:

```
fileprocess.php
```

```
<?php
```

```
$filename = "C:/Documents and
Settings/Administrator/MyFiles/myfile.txt";
```

```
$newfile = fopen($filename, "w+");
```

```
//код для чтения или записи данных в файл располагается здесь
```

```
fclose($newfile);
```

```
?>
```

Первый шаг состоит в создании переменной для хранения всего пути доступа к файлу, который будет открыт:

```
$filename = "C:/Documents and
Settings/Administrators/MyFiles/myfile.txt";
```

Путь доступа к текстовому файлу `myfile.txt` хранится в переменной с именем `filename`. Затем создается указатель файла с именем `$newfile` и применяется с функцией `fopen()` для открытия файла, указанного в предыдущем разделе. Указатель файла является ссылочной переменной PHP, используемой для ссылки на только что открытый файл:

```
$newfile = fopen($filename, "w+");
```

Указатель файла будет использован позже для чтения и записи содержимого в открытый файл.

В некоторых ситуациях функция `fopen()` не сможет успешно открыть файл в результате неверного пути доступа к файлу, полномочий безопасности или других непредвиденных проблем. В связи с этим рекомендуется использовать специальную функцию PHP для аккуратной обработки таких ошибок. Эти функции можно применять в сочетании с оператором управления ошибками PHP "@", рассмотренного в [разделе 8-2](#) для завершения сценария, подавления создаваемых PHP сообщений об ошибках, и вывода более понятного пользователю сообщения. Функции управления ошибками PHP описаны ниже:

`exit(сообщение_об_ошибке)` – завершает текущий сценарий и выводит сообщение об ошибке, передаваемое в функцию.

`die(сообщение_об_ошибке)` – алиас функции `exit()`.

Следующий сценарий демонстрирует использование функций управления ошибками:

`fileprocess.php`

```
<?php
```

```
$filename = "C:/Documents and Settings/Administrator/MyFiles/  
myfile.txt";
```

```
$newfile = @fopen($filename, "w+") or exit("Невозможно открыть  
или создать файл");
```

```
//код для чтения или записи данных в файл размещается здесь
```

```
fclose($newfile);
```

```
?>
```

Если файл невозможно открыть, функция `exit()` выводит сообщение "Невозможно открыть или создать файл" и сценарий завершается.

Если файл открывается успешно, то содержимое можно читать из файла, записывать в файл, или добавлять к файлу в зависимости от используемого в функции `fopen()` режима. Подробнее эти действия будут рассмотрены в следующих разделах. После завершения обработки всего файла применяется функция `fclose()` для закрытия открытого файла.

## Чтение файлов

Этот раздел описывает, как использовать PHP для чтения содержимого файлов в системах Windows.

Для чтения файлов в PHP имеются функции `fread()` и `fsize()`. Они определяются ниже.

`fread(указатель_ресурса, длина)` – функция, используемая для чтения содержимого файла. Читает указанное количество байтов "длина" из файла "указатель\_ресурса". Чтение останавливается, когда будет прочитано заданное количество байтов ("длина") или будет достигнут маркер EOF (end of file). Функция требует два параметра – указатель файла, который создается, когда файл открывается с помощью функции `fopen()`, и размер, который определяет, какая часть содержимого будет считана.

`fgetcsv(указатель_ресурса, длина, ограничитель)` – функция, используемая для чтения содержимого файла и анализа данных для создания массива. Данные разделяются параметром-ограничителем, задаваемым в функции.

`filesize(имя_файла)` – возвращает размер файла. Если возникает ошибка, функция возвращает значение `false`.

Следующий пример иллюстрирует, как прочитать все содержимое файла:

`fileread.php`

```
<?php
```

```
$filename = "C:/Documents and  
Settings/Administrator/MyFiles/myfile.txt";
```

```
$newfile = @fopen($filename, "r") or exit("Невозможно открыть  
файл!");
```

```
$file_contents = @fread($newfile, filesize($filename))  
    or exit("Невозможно прочитать содержимое файла!");
```

```
fclose($newfile);
```

```
?>
```

Первый шаг состоит в создании переменной для хранения полного пути доступа к файлу, который будет открыт для чтения:

```
$filename = "C:/Documents and  
Settings/Administrators/MyFiles/myfile.txt";
```

Путь доступа к текстовому файлу `myfile.txt` хранится в переменной с именем `filename`. Затем создается указатель файла с именем `$newfile` и используется с функцией `fopen()` для открытия файла, указанного в предыдущем разделе. Указатель файла применяется для ссылки на только что открытый файл:

```
$newfile = fopen($filename, "r");
```

Указатель файла является переменной РНР, которая содержит ссылку на открытый файл. Он будет задействован позже с функцией `fread()` для чтения содержимого из открытого файла.

Затем создается переменная с именем `$file_contents` и используется для хранения содержимого текстового файла `myfile.txt`. Первый параметр функции `fread()` указывает на имя файла, содержимое которого будет прочитано. Второй параметр определяет длину файла. Если длина файла неизвестна, можно воспользоваться специальной

функцией PHP с именем `filesize()`, которая определяет длину файла. Она требует один параметр – имя или путь доступа файла, который читается в данный момент.

```
$file_contents = fread($newfile, filesize($filename));
```

Все содержимое текстового файла хранится теперь в переменной `$file_contents`. Эти данные можно выводить на экране с помощью оператора `echo` или записать в другой текстовый файл.

В некоторых случаях может понадобиться прочитать и работать с отдельными частями содержимого текстового файла. При использовании `fread()` все содержимое файла хранится в одной переменной, что затрудняет работу с отдельными частями файла. Если текстовый файл содержит разграничители для разделения отдельных фрагментов данных, можно применить для чтения другую функцию чтения — `fgetcsv()`. Эта функция читает содержимое файла и создает массив, делая доступными определенные части текста.

Предположим, что текстовый файл `numbers.txt` существует и содержит следующие данные:

```
numbers.txt
```

```
50,17,34,90
```

Следующий сценарий демонстрирует использование функции `fgetcsv()` для чтения содержимого текстового файла.

```
fileread.php
```

```
<?php
```

```
$filename = "C:/numbers.txt";
```

```
$newfile = @fopen($filename, "r") or exit("Could not open  
file");
```

```
$file_contents = @fgetcsv($newfile, filesize($filename),",")  
    or exit("Could not read file contents");
```

```
for ($i=0; $i < sizeof($file_contents); $i++)  
{  
    echo $file_contents[$i];  
    echo "<br/>";  
}
```

```
fclose($newfile);
```

```
?>
```



После открытия файла функция `fgetcsv()` считывает все содержимое файла, создавая массив — `'$file_contents'`. Третий параметр функции `fgetcsv()` определяет, что каждый элемент, отделенный с помощью запятой `,`, станет элементом нового массива. Так как `numbers.txt` содержит значения `50,28,34,90`, то `$file_contents[0] = 50`, `$file_contents[1] = 28`, `$file_contents[2] = 34`, `$file_contents[3] = 90`. После создания массива значениями можно манипулировать с помощью любой из функций для массивов PHP. В предыдущем примере цикл `for` выполняет итерации на массиве `$file_contents[]` и выводит каждое число.

После завершения обработки файла функция `fclose()` используется для закрытия открытого файла.

## Запись в файлы

Этот раздел описывает, как использовать PHP для записи содержимого в файлы в системах Windows.

PHP содержит функцию `fwrite()` для записи файлов. Эта функция определена ниже.

`fwrite(указатель_файла, строка)` — записывает содержимое строки в поток указанного файла. Если задан аргумент `length` (длина), запись будет остановлена после записи `length` байтов или достижения конца строки.

Следующий пример иллюстрирует, как записать все содержимое файла:

`filewrite.php`

```
<?php
```

```
$filename = "C:/Documents and  
Settings/Administrator/MyFiles/myfile.txt";
```

```
$newfile = @fopen($filename, "w") or exit("Невозможно открыть  
файл");
```

```
$file_contents = "Добавьте эту строку в текстовый файл";
```

```
fwrite($newfile,$file_contents);
```

```
fclose($newfile);
```

```
?>
```

Первый шаг состоит в создании переменной для хранения полного пути доступа к файлу, который будет открыт или создан:

```
$filename = "C:/Documents and  
Settings/Administrators/MyFiles/myfile.txt";
```

Путь доступа к текстовому файлу `myfile.txt` хранится в переменной с именем `filename`. Затем создается указатель файла с именем `$newfile` и используется с функцией `fopen()` для открытия файла, указанного в предыдущем разделе. Указатель файла применяется для ссылки на только что открытый файл. Файл открывается в режиме записи:

```
$newfile = fopen($filename, "w");
```

Указатель файла является переменной PHP, содержащей ссылку на открытый файл. Он будет задействован позже для записи содержимого в открытый файл.

Затем создается переменная с именем `$file_contents` и ей присваивается строковое значение, которое будет записано в текстовый файл `myfile.txt`.

Наконец вызывается функция `fwrite()`. Первый параметр функции `fwrite()` указывает на имя файла, в который будет записываться содержимое. Второй параметр содержит текст, который будет записываться в открытый файл.

```
fwrite($newfile, $file_contents);
```

После завершения обработки файла используется функция `fclose()` для закрытия открытого файла.

В некоторых случаях может понадобиться записать содержимое существующего файла в новый файл. Этот процесс требует использования функций `fopen()`, `fread()` и `fwrite()`. Открывается первый файл, читается его содержимое и записывается в новый файл, который также был открыт. Следующий сценарий показывает реализацию такого процесса:

```
filecopy.php
```

```
<?php
```

```
$fileAname = "C:/MyFiles/PHP/file1.txt";  
$fileBname = "C:/MyFiles/PHP/file.txt";
```

```
$currentfile = fopen($fileAname,"r") or exit("Невозможно открыть  
файл");  
$fileAcontents = fread($currentfile,filesize($fileAname));
```

```
$newfile = fopen($fileBname,"w");
```

```
fwrite($newfile, $fileAcontents);
```

```
fclose($newfile);
fclose($currentfile);

echo "Содержимое файла file1.txt скопировано в файл file.txt";

?>
```

Этот сценарий копирует содержимое "file1.txt" в новый файл "file.txt". Сначала объявляются две переменные `$fileAname` и `$fileBname`, которым присваиваются пути доступа к каталогам существующего и нового файла. Функция `fopen()` используется для открытия текущего файла для чтения его содержимого. Открытый файл присваивается указателю файла `$currentfile`. Содержимое открытого файла считывается с помощью функции `fread()` и присваивается переменной `$fileAcontents`. Затем функция `fopen()` применяется снова для открытия нового файла. Открытый файл присваивается указателю файла `$newfile`. Функция `fwrite()` используется для записи содержимого исходного файла в новый файл. Когда процесс копирования завершается, оба файла закрываются с помощью функции `fclose()`.

## Копирование файлов

Этот раздел описывает, как использовать PHP для копирования файлов в системах Windows.

PHP содержит функцию `copy()` для копирования файлов. Эта функция определяется ниже:

`copy(исходный_файл, новый_файл)` – копирует содержимое исходного файла, определенного первым параметром, в новый файл, определенный вторым параметром функции. Функция возвращает значение `true` или `false`.

Следующий пример показывает, как скопировать содержимое одного файла в другой файл:

```
filecopy.php

<?php

$orig_filename = "C:/Documents and
Settings/Administrator/MyFiles/myfile.txt";

$new_filename = "C:/Documents and
Settings/Administrator/MyFiles/myNewfile.txt";

$status = @copy($orig_filename, $new_filename) or
die("Невозможно скопировать файл");
```

```
echo "Содержимое файла успешно скопировано!";

?>
```

Первый шаг состоит в создании переменной для хранения полного пути доступа к исходному файлу, содержимое которого будет скопировано:

```
$orig_filename = "C:/Documents and  
Settings/Administrators/MyFiles/myfile.txt";
```

Затем создается вторая переменная для хранения полного пути доступа к новому файлу, который будет создан:

```
$new_filename = "C:/Documents and  
Settings/Administrators/MyFiles/myNewfile.txt";
```

Копирование выполняет функция `copy()`, принимая два параметра, путь доступа к исходному файлу — `$orig_filename`, и путь доступа к новому — `$new_filename`. Функция `copy()` возвращает значение `true`, если копирование завершается успешно; иначе возвращается значение `false`. Возвращаемое значение хранится в переменной `$status`.

```
$status = copy($orig_filename, $new_filename) or die("Невозможно  
скопировать файл");
```

Если функция `copy()` отказывает, выполняется функция `die()`, выводя сообщение об ошибке. Иначе с помощью функции `echo` выводится сообщение об успехе.

```
echo "Содержимое успешно скопировано";
```

В предыдущем разделе функция `fwrite()` используется вместе с функцией `fread()` для чтения содержимого одного файла и записи этого содержимого в новый файл. Если содержимое исходного файла не добавляется к существующему файлу, то функция `copy()` предоставляет более простой подход для копирования содержимого из существующего файла в новый файл.

## Удаление файлов

Этот раздел описывает, как использовать PHP для удаления файлов в системах Windows.

PHP содержит функцию `unlink()` для удаления файлов. Функцию `unlink()` надо использовать с осторожностью. После удаления файла его невозможно восстановить. Эта функция определяется ниже.

`unlink(имя_файла)` – удаляет файл, определенный параметром. Функция возвращает значение `true` или `false`.

Следующий пример показывает, как удалить файл с помощью функции `unlink()`:

`filedelete.php`

```
<?php
```

```
$filename = "C:/Documents and  
Settings/Administrator/MyFiles/myfile.txt";
```

```
$status = unlink($filename) or exit("Невозможно удалить файл");
```

```
echo "файл удален успешно";
```

```
?>
```

Первый шаг состоит в создании переменной для хранения полного пути доступа к файлу, содержимое которого будет удалено:

```
$filename = "C:/Documents and  
Settings/Administrators/MyFiles/myfile.txt";
```

Функция `unlink()` выполняется, получая один параметр, путь доступа исходного файла — `$filename`. Функция `unlink()` возвращает значение `true`, если файл удаляется успешно; иначе возвращается значение `false`. Возвращаемое значение хранится в переменной `$status`.

```
$status = unlink($filename) or die("Невозможно удалить файл");
```

Если функция `unlink()` отказывает, выполняется функция `exit()`, выводя сообщение об ошибке. Иначе с помощью оператора `echo` выводится сообщение об успехе:

```
echo "файл удален успешно";
```

## Переименование файлов

Этот раздел описывает, как использовать PHP для переименования файлов в системах Windows.

PHP содержит функцию `rename()` для переименования файлов. Эта функция определена ниже:

`rename($orig_filename, $new_filename)` – переименует файл, определенный первым параметром, в имя, определенное вторым параметром. Функция возвращает значение `true` или `false`.

Следующий пример показывает, как переименовать файл с помощью функции `rename()`:

filerename.php

```
<?php
```

```
$orig_filename = "C:/Documents and  
Settings/Administrator/MyFiles/myfile.txt";  
$new_filename = "C:/Documents and  
Settings/Administrator/MyFiles/newfile.txt";  
$status = rename($orig_filename, $new_filename) or  
exit("Невозможно переименовать файл");
```

```
echo "файл успешно переименован";
```

```
?>
```

Первый шаг состоит в создании переменной для хранения полного пути доступа к файлу, который будет переименован:

```
$orig_filename = "C:/Documents and  
Settings/Administrators/MyFiles/myfile.txt";
```

Второй шаг состоит в создании переменной для хранения полного пути доступа к файлу, который будет создан, когда старый файл будет переименован:

```
$new_filename = "C:/Documents and  
Settings/Administrators/MyFiles/newfile.txt";
```

При выполнении функция `rename()` получает два параметра, путь доступа к исходному файлу — `$orig_filename` и путь доступа к файлу, который будет создан, когда старый файл будет переименован — `$new_filename`. Функция `rename()` возвращает значение `true`, если файл переименовывается успешно, иначе возвращается значение `false`. Возвращаемое значение хранится в переменной `$status`.

```
$status = rename($orig_filename, $new_filename) or  
exit("Невозможно переименовать файл");
```

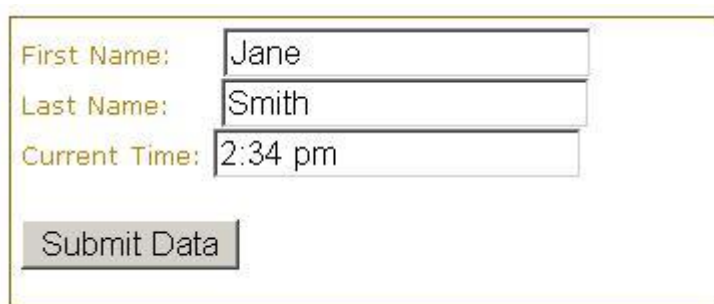
Если функция `rename()` отказывает, выполняется функция `exit()`, выводящая сообщение об ошибке. Иначе выводится сообщение об успехе с помощью оператора `echo`.

```
echo "файл успешно переименован";
```

## Получение данных формы

В большинстве случаев введенные пользователями данные формы записывают в СУБД, такую, как MS Access и MySQL, с помощью функций ODBC и MySQL, рассмотренных в разделах [9](#) и [10](#). Аналогичным образом данные формы можно также записать в текстовый файл. Этот раздел

описывает, как использовать PHP для получения данных формы и записи их в текстовый файл.



First Name: Jane  
Last Name: Smith  
Current Time: 2:34 pm  
Submit Data

Рассмотрим приведенную выше страницу с формой. Следующий пример показывает, как записать отправленные данные формы в текстовый файл:

```
<?php

if ($_POST['SubmitB'] == "Submit Data")
{
$file_name = "c:\formfile.txt";
$open_file = fopen($file_name, "a+");

$file_contents= $_POST['FName'] . "," . $_POST['LName'] . "," .
$_POST['DateTime'] . "\n";

fwrite($open_file,$file_contents);

fclose($open_file);

echo "Данные формы успешно записаны в файл";
}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
<title>Страница Web </title>
</head>
<body>

<p>Запись данных формы в файл </p>

<p>
<form method="post" action="createfile.php">

Enter First Name <input type="text" name="FName"/><br/><br/>
```

```

Enter Last Name <input type="text" name="LName"/><br/><br/>
<input type="hidden" name="DateTime" value="<?php echo date('g:i
a') ?>"/>
<input type="submit" name="SubmitB" value="Submit Data"/>

</form>
</p>

</body>
</html>

```

Рисунок 6.1 – Запись в файл

В этом примере форма страницы XHTML содержит текстовое поле для имени и фамилии пользователя. Также кодируется скрытое поле, в которое заносится с помощью функции PHP `date()` текущие дата и время. Когда нажимается кнопка отправки формы, создается новый текстовый файл `'formfile.txt'` и открывается в режиме добавления:

```

$file_name = "c:\formfile.txt";
$open_file = fopen($file_name, "a+");

```

Затем переменной `$file_contents` присваиваются значения суперглобальных переменных `POST`, содержащие имя и фамилию пользователя и текущее значение даты и времени. К строкам присоединяется запятая, чтобы создать разграничители этих значений. В конце каждой строки добавляется символ новой строки для создания возврата каретки:

```

$file_contents= $_POST['FName'] . "," . $_POST['LName'] . "," .
$_POST['DateTime'] . "\n";

```

Наконец, содержимое переменной `$file_contents` записывается (добавляется) в текстовый файл. Файл закрывается, и используется оператор `echo` для вывода подтверждающего сообщения в окне браузера:

```

fclose($open_file);
echo "Данные формы успешно записаны в файл";

```

## Пересылка файлов

В некоторых динамичных приложениях Web необходимо разрешать пользователям пересылку файлов с локального компьютера на сервер Web. Такое приложение может позволить пользователям обмениваться файлами с другими пользователями или просто предоставить механизм для хранения файлов для будущего использования. Этот раздел вводит функцию PHP для пересылки файлов.

В PHP пересылку файлов можно реализовать с помощью следующей функции:



`move_uploaded_file(имя_файла, место_назначения)` - перемещает файл в указанное место на сервере.

Прежде чем переходить к деталям кода PHP, давайте посмотрим на элементы управления формы XHTML, необходимые для создания страницы для пересылки файлов.

```
<form enctype="multipart/form-data" action="upload.php"
method="post">
```

```
Select File: <input type="file" name="uploadFile">
```

```
<input name="SubmitB" type="submit" value="Upload File">
```

```
</form>
```

Блок кода начинается со стандартного тега XHTML `<form>`. Кроме атрибутов `action` и `method` форма, используемая для пересылки файлов, должна включать тип кодирования или атрибут `"encrypt"`. Когда форма применяется для пересылки файлов должно использоваться значение атрибута `encrypt` `"multipart/form-data"`.

После тега `<form>` следует элемент текстового поля `<input>`. Этот элемент управления нужен для определения расположения и имени файла, который будет пересылаться. Он содержит атрибуты `name` и `type`. Атрибут `type` должен быть задан как `"file"`. `name` имеет определенное пользователем значение, которое будет использоваться сервером для идентификации файла источника во время процесса пересылки.

Текстовое поле `file` включает также кнопку `"Browse..."` при выводе в окне браузера. Когда нажимается эта кнопка, появляется диалоговое окно, позволяющее пользователю найти на своем локальном компьютере файл, который будет пересылаться.

Последним элементом управления является кнопка отправки (`submit`). Кнопка отправки используется для инициирования процесса отправки формы. Когда нажимается эта кнопка, информация о файле посылается на страницу `upload.php`, которая содержит код PHP для пересылки файла в папку, расположенную на сервере Web.

После кодирования формы XHTML на страницу можно добавить сценарий PHP для выполнения динамической пересылки файла. Когда файл пересылается с помощью функции `move_uploaded_file()`, он кратко хранится во временном месте на сервере Web. Для перемещения файла в его конечное место назначения и манипуляций с его различными свойствами, используется суперглобальный массив PHP `$_FILES`. Массив `$_FILES` применяет значение `name`, заданное в теге `<input type="file" name="uploadFile"/>` (в данном случае `'uploadFile'`) для

идентификации пересылаемого файла. Записи, связанные с массивом `$_FILES`, описаны ниже.

`$_FILES['uploadFile']['tmp_name']` – каталог на сервере web, где временно хранится файл. По умолчанию используется каталог `uploadtemp`, расположенный в папке PHP.

`$_FILES['uploadFile']['name']` – имя файла в системе пользователя.

`$_FILES['uploadFile']['size']` – размер файла в байтах.

`$_FILES['uploadFile']['type']` – тип MIME файла.

`$_FILES['uploadFile']['error']` – код ошибки, связанный с пересылкой файла (0 – успешная пересылка, 1 – файл превышает максимальный размер пересылки, 2 – файл превышает максимальный размер файла, 3 – файл частично загружен, 4 – файл не загружен).

Следующие блоки кода показывают, как массив `$_FILES` используется с функцией `move_uploaded_file()` для создания простой процедуры пересылки.

upload.php

```
<?php

    if ($_POST[SubmitB] == "Upload File")
    {

        move_uploaded_file ($_FILES['uploadFile'] ['tmp_name'],
            "../PHPTutorial/ECommerce/Databases/{$_FILES['uploadFile']
['name']}");

        echo "Файл загружен.";

    }

?>

<form enctype="multipart/form-data" action="upload.php"
method="post">

Select File: <input type="file" name="uploadFile">

<input name="SubmitB" type="submit" value="Upload File">

</form>
```

Когда нажимается кнопка "Upload File", то файл, определенный в `<input type="file" name="uploadFile"/>`, автоматически посылается во временную папку на сервере Web. Затем вызывается функция `move_uploaded_file()` для перемещения файла. Первый параметр функции, `$_FILES['uploadFile'] ['tmp_name']`, становится ссылкой на файл, когда функция готовится к перемещению его в конечное место назначения. Второй параметр, `"../PHPTutorial/ECommerce/Databases/{$_FILES['uploadFile'] ['name']}"`, является относительным путем доступа к папке, в которой был временно сохранен файл. Последняя часть пути доступа включает код `{$_FILES['uploadFile'] ['name']}`. Это можно интерпретировать как имя файла, который был введен в текстовое поле `file` с именем "uploadFile". Вкратце, функция `move_uploaded_file()` перемещает файл из временной папки пересылки `folder ($_FILES['uploadFile'] ['tmp_name'])` в папку `"../PHPTutorial/ECommerce/Databases/"`, и файл сохраняется с тем именем, которое ввел пользователь, `({$_FILES['uploadFile'] ['name']})`. Файлы можно пересылать в любой каталог на сервере Web, однако папка места назначения должна иметь полномочия доступа для записи ("write").

Предыдущий пример кода иллюстрирует легкость пересылки файла, но с предположением, что процесс пересылки всегда будет работать, а ошибки никогда не будут происходить. Мы не рассмотрели, что происходит, когда пользователь пытается переслать файл, который превышает ограничение на размер, если папка загрузки не имеет подходящих полномочий безопасности или если некоторые непредвиденные сетевые проблемы мешают пересылке всего файла. Чтобы улучшить приведенный выше код пересылки, необходимо предоставить процедуры, которые проверяют ошибки и предоставляют пользователю информацию о том, как исправить эти проблемы.

Следующие блоки кода показывают модифицированную версию предыдущей процедуры пересылки, которая содержит проверку ошибок.

upload.php

```
<?php
```

```
if ($_POST[SubmitB] == "Upload File")
{
```

```
    move_uploaded_file ($_FILES['uploadFile'] ['tmp_name'],
```

```
    "../PHPTutorial/ECommerce/Databases/{$_FILES['uploadFile']
    ['name']}");
```

```
        if($_FILES['uploadFile'] ['error'] > 0)
        {
            switch ($_FILES['uploadFile'] ['error'])
            {
```

```

        case 1: echo 'Файл превышает максимальный серверный размер
для пересылки';
        break;

        case 2: echo 'Файл превышает максимальный размер файла';
        break;

        case 3: echo 'Файл загрузился только частично';
        break;

        case 4: echo 'Никакой файл не загрузился';
        break;

    }

}

else

{

    echo 'Файл успешно загружен';

}

}

?>

<form enctype="multipart/form-data" action="upload.php"
method="post">

Select File: <input type="file" name="uploadFile"/>

<input type="hidden" name="MAX_FILE_SIZE" value="1000000"/>

<input name="SubmitB" type="submit" value="Upload File"/>

</form>

```

Рисунок 6.2 – Модификация записи в файл

Обновленный код включает оператор `if` и оператор `switch/case` для проверки статуса пересылки файла. После выполнения функции `move_uploaded_file()` оператор `if` проверяет значение массива `$_FILES['uploadFile'] ['error']`. Если значение больше 0, то произошла ошибка. Значение `$_FILES['uploadFile'] ['error']` передается оператору `switch` и проверяется. После этого выводится соответствующее сообщение об ошибке. Если значение `$_FILES['uploadFile'] ['error']` равно 0, оператор `else` выводит сообщение об успешном выполнении.

Еще одним новым свойством, включенным в этот пример, является скрытое текстовое поле XHTML с именем "MAX\_FILE\_SIZE". Это специальный скрытый тег, который можно использовать с тегом `file`, `<input type="file" name="uploadFile"/>`, для задания максимального размера файла. Если файл превышает определенное значение, то произойдет ошибка. Это значение измеряется в байтах. Здесь максимальный размер файла задан равным 1,000,000 байт (приблизительно 1 мегабайт). Можно также задать максимальный серверный размер пересылки. Это максимальный размер файла, заданный на сервере в файле `PHP.ini`.

## 7. Лекция: Отправка E-mail

Эта лекция описывает, как использовать службы SMTP для отправки автоматических сообщений e-mail из приложений PHP

Этот раздел описывает, как использовать службы SMTP для отправки автоматических сообщений e-mail из приложений PHP. E-mail посылается с сервера Web через его службу простого протокола пересылки почты SMTP. Как и предполагает название, это ограниченная в возможностях служба e-mail, однако ее достаточно для создания автоматических сообщений e-mail. Необходимо отметить, что требуется сервер SMTP, чтобы можно было воспользоваться функциями e-mail в PHP. В операционных системах XP Professional, Windows 2000 server и Windows 2003 server службы SMTP объединены с информационными службами Интернет (IIS). В Linux/Unix популярными пакетами SMTP являются Sendmail и Qmail.

При выполнении PHP на сервере с помощью служб SMTP IIS, может понадобиться сконфигурировать его, чтобы разрешить пересылку сообщений e-mail. Выполните следующие действия.

1. Откройте инструменты администрирования IIS
2. Остановите используемую по умолчанию службу виртуального сервера SMTP.
3. Откройте окно свойств используемого по умолчанию виртуального сервера SMTP.
4. Щелкните на вкладке "Access" и нажмите кнопку "Relay...".
5. Нажмите кнопку "Only the list below" и добавьте один компьютер с IP-адресом 127.0.0.1.
6. Нажмите кнопку "OK", чтобы закрыть окно вкладок и свойств "Access".
7. Перезапустите используемую по умолчанию службу виртуального сервера SMTP.

Необходимо также сделать следующие изменения в конфигурационном файле PHP — `php.ini` — чтобы система PHP могла использовать службы SMTP. Откройте файл `php.ini` с помощью текстового редактора и найдите следующие строки:

```
[mail function]
;For Win32 only
SMTP = localhost
;For Win 32 only
sendmail_from = me@localhost.com
```

Необходимо изменить директиву SMTP, чтобы она указывала на используемый сервер SMTP. Если используются локальные службы SMTP, то это значение должно быть задано как `localhost`. Вторая директива `sendmail_from` является адресом email, применяемым в заголовке `From` исходящей почты e-mail. Должна быть задана действительная учетная запись e-mail, если пользователям будет разрешено отвечать на автоматически создаваемые сообщения e-mail.

В PHP имеется функция `mail()` для отправки e-mail. Эта функция определена ниже:

```
mail(string_to, string_subject,
string_message,string_additional_headers) – позволяет посылать
сообщение e-mail. Возвращает true, если сообщение успешно послано,
иначе возвращается значение false.
```

Следующий пример демонстрирует использование функции `mail()`:

```
<?php

$to      = 'youraddress@domain.com';
$subject = 'the subject';
$message = 'hello';
$headers = 'From: webmaster@example.com' . "\r\n" .
    'Reply-To: webmaster@example.com' . "\r\n" .
    'X-Mailer: PHP/' . phpversion();

mail($to, $subject, $message, $headers);

?>
```

Первый шаг состоит в создании переменной для хранения адреса e-mail, куда будет послано сообщение:

```
$to = "youraddress@domain.com";
```

Это может быть любой действительный адрес e-mail. Несколько адресов e-mail должны разделяться запятой ",".

Переменная `$subject` содержит тему сообщения e-mail. Эта строка появится в строке `subject` (тема) сообщения.

```
$subject="PHP Mail";
```

Основное содержание тела сообщения e-mail присваивается переменной `$msg`. Если потребуется, то можно соединять несколько переменных `$msg` вместе. Это часто бывает нужно, когда посылается длинное описательное сообщение.

```
$msg = "Сообщение, созданное с помощью функции PHP mail().";
```

Затем создаются заголовки e-mail и присваиваются переменной `$headers`. Заголовки e-mail являются строками в начале сообщений e-mail, которые определяют их структуру и делают их, по сути, действительными почтовыми адресами. Хотя функция `mail()` может использоваться без заголовков, рекомендуется включать заголовки "From:" и "Reply-To:"

```
$headers = "From: My Web Site <myaddress@mydomain.com>";  
$headers .= "Reply-To: myaddress@mydomain.com";
```

Наконец, вызывается функция `mail()` для отправки сообщения:

```
mail($to,$subject,$msg, $headers);
```

В большинстве случаев параметры `to`, `subject`, и `message` функции `mail()` не кодируются жестким образом, как показано в предыдущем примере. Вместо этого они подставляются динамически в результате ввода пользователя. Например, рассмотрим страницу, которая позволяет пользователю электронным образом регистрироваться для получения товара или услуги. Пользователь вводит имя, фамилию, адрес e-mail, и номер телефона. Эта информация передается на страницу PHP, которая анализирует информацию и посылает пользователю подтверждающее сообщение e-mail. Следующий пример демонстрирует этот процесс:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"  
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"  
lang="en">
```

```
<head>
```

```
  <title>Страница Web </title>
```

```
</head>
```

```
<body>
```

```
<h3> Страница регистрации </h3>
```

```
<form name="registration" method="post" action="email.php">
```

```
First Name: <input type="text" name="fname"/>
```

```
Last Name: <input type="text" name="lname"/>
```

```
Email Address: <input type="text" name="email"/>
```

```
Telephone: <input type="text" name="telephone"/>
```

```
<input type="submit" name="Submit Registration"/>
```

```
</form>

</body>
</html>
```

Страница `registration.htm` является стандартной страницей формы XHTML, которая позволяет пользователю вводить имя, фамилию, адрес e-mail и номер телефона. Когда нажимается кнопка "Submit Registration", данные формы передаются на страницу PHP `email.php` как переменные PHP `$_POST[]`:

`$_POST['fname']` – содержит имя пользователя

`$_POST['lname']` – содержит фамилию пользователя

`$_POST['email']` – содержит адрес e-mail пользователя

`$_POST['telephone']` – содержит номер телефона пользователя

Следующий сценарий показывает, как информация формы анализируется и используется функцией `mail()`:

```
<?php

$to = $_POST[email];
$subject = "Подтверждение регистрации";
$msg = "Дорогой: " . $_POST[fname] . " " . $_POST[lname] .
", \n\n";
$msg .= "Вы успешно зарегистрировались.";

$headers = "From: Registration Site <myaddress@mydomain.com>";
$headers .= "Reply-To: registration@mydomain.com";

mail($to, $subject, $msg, $headers);

?>
```

Страница `email.php` получает значения из суперглобального массива `$_POST[]` (содержащего значение, отправленные из `registration.htm`) и присваивает их скалярным переменным, с которыми будет проще работать. Затем создаются заголовки e-mail и вызывается функция `mail()`. Скалярные переменные передают функции необходимые параметры.