Федеральное агентство по образованию

ГОУ ВПО «Шадринский государственный педагогический институт»

Кафедра программирования и сетевых технологий

Курсовая работа

«Разработка веб файлового менеджера с использованием технологии Ajax»

студента 389 группы

*Аллахвердиева Фармана*

руководитель

*Кириллов А. Г.*

Шадринск 2008

**Содержание**

Введение

Технология Ajax.

Преимущества Ajax

Системы, созданные с использованием Ajax

Базовые технологии Ajax

Файловый веб менеджер на основе технологии Ajax

Основные возможности

Принцип реализации программного кода

Заключение

**Введение**

Для реализации своего веб менеджера я решил использовать технологию Ajax как наиболее актуальную, и в настоящее время получившую наибольшее распространение в клиент серверных приложениях, например таких известных веб ресурсах как Google Maps и Gmail

Преимущество Ajax состоит в том, что данная технология позволяет обрабатывать более обширный набор событий, соответствующих действиям пользователя. Становится возможным реализовать перетаскивание объектов и другие сложные функции интерфейса, в результате чего Web - приложение становится похожим на обычную прикладную программу, выполняемую на настольной системе. С точки зрения практичности эта свобода действий важна не потому, что она больше соответствует представлениям пользователя о виртуальном мире, а потому, что позволяет лучше сочетать взаимодействие с пользователем и запросы к серверу.

**Технология Ajax**

**Преимущества Ajax**

Браузер имеет дело с приложением, а не с содержимым для классического приложения на базе Web-страниц браузер представляет собой лишь низкоуровневый терминал. Он не имеет информации о том, какой этап работы выполняется пользователем. На сервере содержатся минимальные сведения об этом, которые, по сути, сводятся к поддержке сеанса.

Если вы работаете с Java или .NET, средства поддержки сеанса на сервере доступны, подобно запросам ответам и MIME-типам, посредством стандартного API.

Когда пользователь регистрируется или другим способом инициализирует сеанс, создается несколько объектов на стороне сервера. Они представляют, например, "корзинку" покупателя или платежную карточку пользователя. Одновременно браузер получает исходную страницу. Она доставляется в виде потока HTML-данных, которые представляют собой сочетание стандартных элементов и данных, специфических для конкретного пользователя. При каждом обращении к серверу браузер получает очередную страницу, содержащую данные тех же типов, что в предыдущих документах. Браузер исправно убирает с экрана старый документ и отображает новый. Других действий от него и не следует ожидать: это низкоуровневая программа, которая делает только то, что было предусмотрено разработчиками. Когда пользователь активизирует ссылку, соответствующую окончанию сеанса, или закрывает браузер, выполнение приложения завершается и сеанс разрушается. Информация, которую пользователь должен увидеть при следующей регистрации, заносится в долговременное хранилище. В Ajax - приложении часть прикладной логики переносится на браузер.

После регистрации пользователя клиентское приложение доставляется браузеру. На многие действия пользователя это приложение способно реагировать самостоятельно. Если имеющихся в наличии возможностей недостает, оно передает запросы серверу, не прерывая последовательность действий пользователя.

При регистрации пользователя браузеру предоставляется более сложный документ, существенную часть которого составляет код JavaScript. Этот документ остается доступным пользователю в течение всего сеанса; при этом, в зависимости от действий пользователя, он изменяет свой внешний вид. Клиентская программа знает, как реагировать на вводимые данные, и способна решать, обрабатывать ли их самостоятельно, посылать ли запрос серверу (который в свою очередь обратится к базе данных или к другому ресурсу) или сделать и то и другое.

Поскольку документ присутствует на стороне клиента в течение всего сеанса, он способен хранить информацию о его состоянии. Например, сведения о состоянии "корзинки" покупателя могут храниться не на сервере, а в клиентской программе.

Как было сказано ранее, на каждом этапе работы классическое Web - приложение предоставляет сочетание стандартных элементов и специальных данных. Когда пользователь добавляет товар в "корзинку", приложение в ответ должно лишь изменить общую цену либо, при наличии ошибки, сообщитьо ней. Код, отвечающий за эти действия, составляет незначительную часть документа. В Ajax-приложении "корзинка" может обладать более высоким "интеллектом" и передавать серверу асинхронные запросы. Шаблон, элементы навигации и другие компоненты страницы уже присутствуют на стороне клиента, поэтому сервер должен передавать только данные, полученные в результате обработки запроса.

Ajax-приложение может достичь данной цели различными способами, например, вернуть фрагмент JavaScript-кода, поток, содержащий обычный текст или небольшой XML-документ. Достаточно заметить, что данные в любом из этих форматов будут иметь значительно меньший объем, чем страница, возвращаемая классическим Web-приложением.

Трафик при работе Ajax-приложениями, сначала возрастает так как при регистрации пользователя клиентской программе доставляется сложное приложение. Дальнейшее взаимодействие с сервером становится гораздо более эффективным.

Для переходного приложения, созданного на базе Web-страниц, суммарный трафик может оказаться меньше, чем для Ajax-приложения. Однако по мере увеличения времени работы пользователя ситуация меняется и Ajax приложение становится гораздо более экономичнее своего конкурента, выполненного в рамках классического подхода.

В Web-браузере предусмотрены два основных механизма ввода данных: гипертекстовые ссылки и HTML-формы. Гипертекстовые ссылки могут быть сформированы на сервере и снабжены параметрами CGI (Common Gateway Interface — интерфейс общего шлюза). Их можно оформить как изображения и средствами CSS (Cascading Style Sheets — каскадные таблицы стилей) организовать обратную связь с пользователями, например, обеспечить изменение внешнего вида при наведении на них курсора мыши. Хороший Web-дизайнер при желании добьется того, что ссылки будут выглядеть как полноправные компоненты пользовательского интерфейса.

Формы могут содержать многие компоненты, типичные для пользовательского интерфейса обычных приложений, а именно: поля редактирования, флажки и переключатели опций, раскрывающиеся списки и пр. Однако некоторые из компонентов в составе форм не поддерживаются. Так, например, в формах не предусмотрены деревья и таблицы. Формы, как и гипертекстовые ссылки, содержат URL, указывающие на ресурсы сервера. Гипертекстовые ссылки и формы могут также указывать на функции JavaScript. В традиционных Web-документах часто можно встретить JavaScript-сценарии, проверяющие корректность заполнения форм. Они следят за незаполненными полями, значениями, выходящими за пределы допустимого диапазона, и другими подобными ошибками. Передача данных на сервер происходит лишь в том случае, если форма заполнена корректно. JavaScript-функции присутствуют на стороне клиента в течение того же времени, что и содержащая их Web-страница.

При обращении к очередной странице пользователю приходится на время прервать работу. Предыдущий документ еще некоторое время отображается на экране; некоторые браузеры даже позволяют активизировать какую-либо из видимых ссылок, но результаты предсказать невозможно. Скорее всего, пользователь, поступивший подобным образом, разрушит информацию о сеансе, поддерживаемую на сервере. После получения новой страницы пользователю предоставляются приблизительно те же возможности выбора, которые он имел до этого. Например, добавление в "корзинку" товара, скажем, брюк, вряд ли приведет к переходу от категории "мужская одежда" к категории "женская одежда" или "детская одежда". Представим себе теперь, как ведет себя "корзинка" покупателя, реализованная в составе Ajax-приложения. Поскольку Ajax позволяет передавать данные в асинхронном режиме, пользователь может добавлять товар в "корзинку" с той же скоростью, с которой он щелкает мышью. Если клиентская часть приложения выполнена профессионально, она без труда справится с данной задачей, и пользователю не придется в своей работе учитывать специфику взаимодействия программы и сервера. Очевидно, что реальной "корзинки" не существует; она выполнена в виде объекта поддержки сеанса на сервере. Однако пользователь, который делает покупки, не должен ничего знать об объекте сеанса. "Корзинка" — это понятие, позволяющее упрощенно представить выполняемые действия. Необходимость переходить от понятия "корзинка" к объектам, содержащимся в памяти компьютера, создает дискомфорт для пользователя. Необходимость ожидать обновления страницы принуждает вернуться из виртуального мира в реальный. Приложение, реализованное средствами Ajax, свободно от этого недостатка. Приобретать товары в сетевом магазине пользователям приходится лишь время от времени, однако в других областях деятельности, например, при решении сложных инженерных задач, поминутное прерывание работы, связанное с необходимостью ожидать появления очередной страницы, недопустимо.

Еще одно преимущество Ajax состоит в том, что данная технология позволяет обрабатывать более обширный набор событий, соответствующих действиям пользователя. Становится возможным реализовать перетаскивание объектов и другие сложные функции интерфейса, в результате чего Web - приложение становится похожим на обычную прикладную программу, выполняемую на настольной системе. С точки зрения практичности эта свобода действий важна не потому, что она больше соответствует представлениям пользователя о виртуальном мире, а потому, что позволяет лучше сочетать взаимодействие с пользователем и запросы к серверу.

Для того чтобы обратиться к серверу в классическом Web-приложении, мы должны щелкнуть на гипертекстовой ссылке либо активизировать элемент формы, а затем ожидать результата. Это неизбежно отвлекает от основной работы. Если же обращение к серверу происходит в ответ на перемещение мыши, перетаскивание объекта или нажатие клавиши, сервер работает параллельно с пользователем. В данном примере приложение реагирует на нажатие клавиш по мере того, как пользователь вводит информацию в поле. Клиент взаимодействует с сервером, извлекает и отображает наиболее вероятные завершения фраз.

В классических Web-приложениях для реализации дополнительных возможностей в документы часто включают фрагменты JavaScript-кода. Страничная модель не позволяет этим фрагментам присутствовать на стороне клиента дольше самой страницы, а это ограничивает их применимость. В результате JavaScript приобрел репутацию языка, пригодного лишь для поделок и незаслуживающего внимания серьезных разработчиков.

Создание кода для Ajax-приложений существенно отличается от того, к чему успели привыкнуть программисты. Код, доставляемый клиенту в начале работы с приложением, доступен до завершения сеанса, причем он непрерывает и не замедляет работу пользователя и не приводит к напрасному расходованию памяти. Если разработчик стремится выйти на рынок независимых приложений, то он должен учитывать, что его программа будет использоваться по нескольку часов подряд. Для этого нужен высокопроизводительный код, допускающий сопровождение. При его создании необходима дисциплина программирования и понимание задачи, как и при написании серверных программ.

Объем кода, вероятно, будет большим, чем в случае классического Web -приложения. В результате большее значение получает сама структура кода. Создание приложения уже становится не под силу одиночке — этим должен заниматься коллектив разработчиков. В процессе работы необходимо обеспечить выделение подзадач, учитывать дальнейшую поддержку, одним словом, работа будет происходить по тому же сценарию, что и создание обычных программ для настольных систем.

Ajax уже используется для создания реальных приложений, и преимущества данного подхода видны на практике.

## Системы, созданные с использованием Ajax

Наибольший вклад в формирование современного представления об Ajax - приложениях внесла компания Google. (Она использовала данную технологию еще до того, как та получила имя Ajax.) Бета-версия службы GMail стала доступна в начале 2004 года. Эта служба привлекла внимание пользователей не только размерами предоставляемого им почтового ящика, но и интерфейсом, который позволял одновременно открывать несколько сообщений и автоматически обновлял список корреспонденции, даже если пользователь подготавливал в это время новое сообщение. Это был существенный шаг вперед по сравнению с обычными почтовыми системами, предлагаемыми большинством провайдеров. Сравнивая GMail с Web-интерфейсами корпоративных почтовых серверов, например Microsoft Outlook и Lotus Notes, нетрудно заметить, что GMail обеспечивает большинство функций, не прибегая к помощи тяжеловесных и ненадежных элементов ActiveX или Java-аплетов. В результате служба доступна не только для корпоративных пользователей, вооруженных специально настроенными машинами, но и для большинства обычных систем.

За GMail последовали другие интерактивные службы, например, Google Suggest, поисковый сервер которой автоматически предлагает завершение фразы, указываемой в составе запроса, и Google Maps — интерактивная масштабируемая карта, посредством которой определяется расположение ресурса. Начали эксперименты с данной технологией и другие компании. В качестве примера можно привести интерактивную систему Flickr, которая в настоящее время является составной частью Yahoo!.

Упомянутые выше системы — лишь проверка возможностей нового подхода, они представляют собой переходные приложения, применяемые лишь эпизодически. За ними последовали независимые Ajax-приложения.

Наличие приложений, созданных на базе Ajax, свидетельствует о том, что новый подход получает признание разработчиков. Если отдельные программисты иногда используют новую технологию лишь для того, чтобы ознакомиться с ней, то такие компании, как Google и Yahoo!, прибегают к ней только в том случае, если она сулит выигрыш в конкурентной борьбе.

Можно ожидать, что с ростом потребности в высококачественном прикладном интерфейсе традиция использования Ajax получит дальнейшее развитие. Вполне вероятно, что в течение ближайших нескольких лет позиции Ajax-приложений на рынке будут укрепляться.

**Базовые технологии Ajax**

Ajax — не одна конкретная технология, скорее это совокупность четырех технологий, дополняющих друг друга (табл.1).

Сложные Ajax-приложения доставляются пользователям и как пользователи взаимодействуют с ними за счет использования JavaScript-кода, который "объединяет" приложение, определяет последовательность действий пользователя и бизнес-логику программы. Действия с интерфейсом преобразуются в операции с элементами DOM (Document Object Model), с помощью которых обрабатываются данные, доступные пользователю, в результате чего представление их изменяется. Здесь же производится обработка перемещений и щелчков мышью, а также нажатий клавиш. Каскадные таблицы стилей, или CSS (Cascading Style Sheets), обеспечивают согласованный внешний вид элементов приложения и упрощают обращение к DOM-объектам. Объект XMLHttpRequest (или подобные механизмы) используется для асинхронного взаимодействия с сервером, обработки запросов пользователя и загрузки в процессе работы необходимых данных. Три из этих четырех технологий — CSS, DOM и JavaScript — составляют DHTML (Dynamic HTML). Следует заметить, что средства DHTML, появившиеся в 1997 году, подавали большие надежды, но так и не оправдали их. DHTML позволяет создавать на базе Web-страниц интерфейсы с достаточно большими интерактивными возможностями, но любые изменения внешнего вида страницы реализуются лишь путем повторной загрузки всего документа. Набор действий, которые можно выполнить без обращения к серверу, весьма ограничен. Средства DHTML интенсивно используются в Ajax, но благодаря асинхронным запросам можно получить результаты, которые невозможно получить с помощью обычных Web-страниц.

Таблица 2.1. Базовые технологии Ajax

|  |  |
| --- | --- |
| JavaScript | Это язык сценариев общего назначения, предназначенный для включения кода в Web-приложение. Интерпретатор JavaScript в составе Web-браузера обеспечивает взаимодействие со встроенными средствами браузера. Данный язык используется для создания Ajax-приложений |
| CSS (Cascading Style Sheets) | CSS предоставляет возможность определять стили  элементов Web-страницы. С помощью данной технологии можно без труда обеспечить согласованность внешнего вида компонентов приложения. В Ajax CSS используется для изменения представления интерфейса в процессе интерактивного взаимодействия |
| DOM (Document Object Model) | DOM представляет структуру Web-страницы в виде набора объектов, которые можно обрабатывать средствами JavaScript. Это дает возможность изменять внешний вид интерфейса Ajax-приложения в процессе работы |
| Объект XMLHttpRequest | Объект XMLHttpRequest позволяет программисту  получать данные с Web-сервера в фоновом режиме. Как правило, возвращаемая информация предоставляется в формате XML, но данный объект позволяет также работать с любыми текстовыми данными |

Очень важно, что средства поддержки всех рассматриваемых здесь технологий уже присутствуют в большинстве современных браузеров, включая Microsoft Internet Explorer, семейство Mozilla/Gecko, Firefox, Mozilla Suite, Netscape Navigator, Camino, Opera, Apple Safari и Konqueror (который ориентирован на выполнение в среде Unix KDE). К сожалению, конкретные реализации этих технологий в разных браузерах различаются рядом важных деталей, более того, различия встречаются даже в разных версиях одного продукта. Правда, за последние пять лет положение дел несколько улучшилось, и разработаны способы, позволяющие справиться с несовместимостью браузеров.

В составе каждой современной операционной системы имеется браузер. Таким образом, подавляющее большинство настольных и портативных компьютеров уже готово для запуска Ajax-приложений. Разработчики программ на базе Java или .NET могут лишь мечтать об этом. Браузеры для КПК и мобильных телефонов обычно имеют усеченный набор возможностей и не поддерживают полный набор Ajax-технологий. Следует заметить, что если бы средства поддержки Ajax и имелись в наличии, все равно размеры экрана и особенности ввода данных создавали бы существенные проблемы при работе с Ajax-приложениями. На сегодняшний день инфраструктура Ajax ориентирована лишь на настольные и портативные компьютеры.

**Файловый веб менеджер на основе технологии Ajax**

**Основные возможности**

Данный файловый веб менеджер предназначен для работы с файлами по сети, используя стандартные программы для просмотра интернет страниц, например Explorer или Firefox.

К основным возможностям программы относится:

* просмотр;
* редактирование;
* копирование;
* переименование;
* перенос;
* создание каталога;
* создание файла;
* удаление.

**Принцип реализации программного кода**

Файловый веб менеджер написанный на Ajax работает на стороне сервера, что требует наличия для его работы, сервер работающий с Php скриптами. Данная программа является кросс платформенной, что позволяет использовать её как на NT платформах, так и на Unix системах.

Основным языком программирования для написания данного менеджера, является Php.

**Функция реализации просмотра файлов**

*<?*

*function copy\_dir($src,$dst) //копирование каталога с поддерикториями*

*{*

*if(!file\_exists($src)) return false; //если файла не существует, выходит из функции*

*if(is\_file($src)) return @copy($src,$dst); //если файл производит его копирование и фозвращает результат копирования*

*$error=false;*

*if(!mkdir($dst)) $error=true; //создает требуемый каталог*

*$dh=@opendir($src); //открывает каталог, который необходимо скопировать*

*while(false!==($f=readdir($dh))) { // читает послеовательно содерживое каталога*

*if($f=='.' || $f=='..') continue; //перехрдит на следующую итерацию, если каталог . или ..*

*if(!copy\_dir($src.'/'.$f,$dst.'/'.$f)) $error=true;*

*}*

*@closedir($dh);*

*return !$error;*

*}*

*function TryCopy($src,$dst,$name)*

*{*

*return copy\_dir($src,$dst."/".$name);*

*}*

*function doView()*

*{*

*global $FS;*

*$fname=$FS.$\_POST["data"]["fname"];*

*if (!file\_exists($fname)) {echo "ViewAnswer(2);";return false;}*

*if (($text=@file\_get\_contents($fname))===false) echo "ViewAnswer(1);";*

*else {*

*$text=preg\_replace("/\"/","\\\"",$text);*

*echo "text=\"".mysql\_escape\_string($text)."\"; ViewAnswer(0)";*

*}*

*}*

*doView();*

*?>*

**Функция реализации редактирования и создания файла**

*<?*

*function Save($name) // сохранение файла*

*{*

*if (!($f=@fopen($name,"w"))) return 1; //попытка открытия файла для записи*

*fputs($f,$\_POST["data"]["text"]); //запись в файл*

*fclose($f); //закрытие файла*

*return 2;*

*}*

*function doEdit()*

*{*

*global $FS;*

*$saved=0;*

*$name=$FS.$\_POST["data"]["name"];*

*if (@$\_POST["data"]["save"]) $saved=Save($name);*

*if (!file\_exists($name)) {echo "EditAnswer(1,$saved);";return false;}*

*if (($text=@file\_get\_contents($name))===false) echo "EditAnswer(1,$saved);";*

*else*

*echo "text=\"".mysql\_escape\_string($text)."\";EditAnswer(0,$saved);";*

*}*

*doEdit();*

*?>*

**Функция реализующая копирование**

*<?*

*function copy\_dir($src,$dst) //копирование каталога*

*{*

*if(!file\_exists($src)) return false; //выход из функции если копируемого файла/каталога не существует*

*if(is\_file($src)) return @copy($src,$dst); // если необходимо скопировать файл, производим копирование и возвращаем результат копирования(ошибка или нет)*

*$error=false;*

*if(!mkdir($dst)) return false; //если не удалось создать каталог выходим их функции*

*$dh=@opendir($src); // открываем каталог для работы*

*while(false!==($f=readdir($dh))) { // цикл выполняется пока в каталоге не перебранный все файлы*

*if($f=='.' || $f=='..') continue; //если файл/каталог - . или .. переходим на следующую итерацию*

*if(!copy\_dir($src.'/'.$f,$dst.'/'.$f)) $error=true; // вызываем эту же функцию, передав имя копируемого файла/каталога и имя результирующего каталога*

*}*

*@closedir($dh); //закрываем открытый каталог*

*return !$error;*

*}*

*function TryCopy($src,$dst,$name)*

*{*

*return copy\_dir($src,$dst."/".$name); // вызываем функцию копирования*

*}*

*function doCopy()*

*{*

*global $FS;*

*$src=$FS.$\_POST["data"]["source"]; //считываем исходное имя каталога/файла из данных, пришедших с сервера*

*$dst=$FS.$\_POST["data"]["dst"]; //считываем каталог в который необходимо скопировать*

*$name=$\_POST["data"]["name"]; //считываем нового имя источника копирования*

*if (!file\_exists($dst)) {echo "CopyAnswer(3);";return false;}// проверяем существование каталога в который необходимо скопировать*

*if (!file\_exists($src)) {echo "CopyAnswer(2);";return false;} // проверяем существование источника копирования*

*if (!TryCopy($src,$dst,$name)) echo "CopyAnswer(1);"; // выполняем копирование и возвращаем в качестве результата вызов*

*else echo "CopyAnswer(0)"; // функции с определенны статусом завершения копирования*

*}*

*doCopy();*

*?>*

**Функция реализующая перенос и переименование.**

*<?*

*function TryRename($old,$new)*

*{*

*return @rename($old,$new);//перемименование каталога/фалйа*

*}*

*function doRename()*

*{*

*global $FS;*

*$old=$FS.$\_POST["data"]["oldname"];*

*$new=$FS.$\_POST["data"]["newname"];*

*if (!file\_exists($old)) {echo "RenameAnswer(2);";return false;}*

*if (!TryRename($old,$new)) echo "RenameAnswer(1);";*

*else echo "RenameAnswer(0)";*

*}*

*doRename();*

*?>*

**Функция реализующая создание каталога**

*<?*

*function doCreate()*

*{*

*global $FS;*

*$src=$FS.$\_POST["data"]["dir"];*

*$name=$\_POST["data"]["name"];*

*if (!file\_exists($src)) {echo "CreateDirAnswer(2);";return false;} //проверяет существование файла/каталога с таким же именем*

*if (!@mkdir($src."/".$name)) echo "CreateDirAnswer(1);"; //создает каталог*

*else echo "CreateDirAnswer(0)";*

*}*

*doCreate();*

*?>*

**Функция для удаления каталога или файла**

*<?*

*function del\_dir ($path)*

*{*

*$dir = @opendir($path);//открытие каталога*

*while(($fname = readdir($dir))){ // цикл, осуществляющий проссмотр содержимого файла*

*if (($fname==".") or($fname=="..")) continue;//переход на следующую итерацию, если имя файла/каталога - . или ..*

*if ( is\_file ($path."/".$fname)) @unlink ($path."/".$fname);//если просмотриваемый элемент - файл, удаляем его*

*else if (!del\_dir ($path."/".$fname)) return false; //если просмотриваемый элемент - каталог,вызываем эту же функцию*

*}*

*return @rmdir($path);//удаляем каталог и возвращаем результат удаления*

*}*

*function TryDelete($path)*

*{*

*if (is\_dir($path)) return del\_dir($path);*

*else return @unlink($path);*

*}*

*global $FS;*

*$path=$FS.$\_POST["data"]["delname"];*

*if (!file\_exists($path)) echo "DeleteAnswer(2);";*

*if (!TryDelete($path)) echo "DeleteAnswer(1);";*

*else echo "DeleteAnswer(0)";*

?>

**Заключение**

Я считаю что выполнил основное назначение работы.В ходе проделанной работы я получили представление о технологии Ajax , наглядно продемонстрировал её возможности и преимущества по сравнению с другими технологиями.

При проектировании своего веб менеджера, я изучил возможности серверных программ поддерживающих работу с Php скриптами, возможности языка программирования Php и Java скриптов.