Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра электронных приборов (ЭП)

Е.С. Шандаров

Информационные системы
Лабораторный практикум

Учебно-методическое пособие

Томск 2007

Введение

Лабораторный практикум по курсу “Информационные системы” включает в себя четыре лабораторных работы. В процессе выполнения заданий студенты создают прототип информационной системы с использованием web-технологий.

Краткий справочник по языку HTML

HTML был изобретён в 1990 году учёным Тимом Бёрнсом-Ли (Tim Berners-Lee) и предназначался для облегчения обмена документами между учёными различных университетов. Проект имел больший успех, чем Tim Berners-Lee мог ожидать. Этим изобретением HTML он заложил основы современной сети Internet.

HTML это язык, который позволяет представлять информацию (например, научные исследования) в Internet. То, что вы видите при просмотре страницы в Internet, это интерпретация вашим браузером HTML-текста. Чтобы увидеть HTML-коды страницы в Internet, щёлкните "View" в линейке меню вашего браузера и выберите "Source".

Структура HTML-документа

Для того, чтобы текстовый файл превратился в HTML-файл, поменять его расширение с ".txt" на ".html" недостаточно. Надо соблюсти "правило первой строки":

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, обязан начинаться со строки декларации версии HTML !DOCTYPE, которая обычно выглядит так:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

Эта строка поможет браузеру определить, как правильно интерпретировать данный документ. В данном случае мы говорим браузеру, что HTML соответствует международной спецификации версии 3.2, которая хоть и не отлича-

ется новизной, но, в отличие от более поздних версий, является полноценным, широко распространенным стандартом без каких-либо неопределенностей.

После объявления версии и типа документа необходимо обозначить его начало и конец. Это делается с помощью тега-контейнера <html>. Необходимо отметить, что любой HTML-документ открывется тегом <html> и им же закрывается.

Затем, между тегами <html> и </html> следует разместить заголовок и тело документа. Вот как должен выглядеть ваш базовый HTML-файл перед началом работы:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Заголовок документа</title>
</head>
<body>
Текст документа
</body>
</html>
```

Из схемы видно, что документ состоит из двух основных блоков – "заголовка" и "тела документа". Заголовок определяется с помощью элемента HEAD, а тело – элементом BODY.

Заголовок содержит "техническую" информацию о документе, хотя чаще всего используется только для обозначения его названия (см. элемент TITLE).

Тело документа – святая святых. Именно в нем находится все то, что отображается на странице: текст, картинки, таблицы. Соответственно, делаем вывод: большинство ваших HTML-экспериментов будет проводиться в пространстве между тегами <BODY> и </BODY>.

Заголовок HTML-документа

Заголовок документа создается с помощью элемента HEAD, между тегами которого размещаются элементы, содержащие техническую информацию о документе. Заголовок обычно располагается до тела документа (см. структуру HTML-документа).

Элементы, относящиеся к заголовку документа:

- HEAD Определяет начало и конец заголовка документа
- TITLE Определяет имя всего документа, которое отображается в заголовке окна браузера
- BASE Определяет базовый адрес, от которого отсчитываются относительные линки внутри документа
- STYLE Используется для вставки в документ таблицы стилей CSS
- LINK Описывает взаимосвязь документа с другими объектами
- META Используется для вставки метаданных

HEAD

Определяет начало и конец заголовка документа. Является контейнером для элементов, содержащих техническую информацию о документе. (TITLE,BASE,STYLE,LINK,META).

Пример:

```
<HTML>
<!-- Начинаем заголовок... -->
<HEAD>
<title>Справочник по HTML</title>
</HEAD>
<!-- ...кончили. Дальше пошло тело документа -->
<BODY>
Текст документа
</BODY>
```

</HTML>

TITLE

Определяет имя всего документа. Имя, как правило, отображается в заголовке окна браузера. Данный элемент обязателен для любого HTML-документа и может быть указан не более одного раза.

Пример:

...

<HEAD>

<TITLE>Руководство по эксплуатации</TITLE>

</HEAD>

BODY

Указывает начало и конец тела HTML-документа. Между начальным и конечным тегами содержится текст документа, изображения и таблицы. Одним словом, все HTML-элементы, отвечающие за отображение документа, управление им и гипертекстовые ссылки. Элемент BODY должен встречаться в документе не более одного раза.

Атрибуты:

MARGINHEIGHT – определяет ширину (в пикселах) верхнего и нижнего полей документа. Работает только в браузерах Netscape.

TOPMARGIN – определяет ширину (в пикселах) верхнего и нижнего полей документа. Работает только в браузерах Internet Explorer.

MARGINWIDTH – определяет ширину (в пикселах) левого и правого полей документа. Работает только в браузерах Netscape.

LEFTMARGIN – определяет ширину (в пикселах) левого и правого полей документа. Работает только в браузерах Internet Explorer.

BACKGROUND – определяет изображение для "заливки" фона. Значение задается в виде полного URL или имени файла с картинкой в формате GIF или

JPG.

BGCOLOR – определяет цвет фона документа.

TEXT – определяет цвет текста в документе.

LINK – определяет цвет гиперссылок в документе.

ALINK – определяет цвет подсветки гиперссылок в момент нажатия.

VLINK – определяет цвет гиперссылок на документы, которые вы уже просмотрели.

Значения атрибутов BGCOLOR, TEXT, LINK, ALINK и VLINK задаются либо RGB-значением в шестнадцатиричной системе, либо одним из 16 базовых цветов.

Замечания: несмотря на то что с помощью атрибутов тега BODY можно задать много полезных параметров HTML документа, в настоящий момент целесообразнее использовать CSS – каскадные таблицы стилей.

Пример:

```
<HTML>
<BODY BACKGROUND="images/bricks.jpg" BGCOLOR="#202020"
TEXT="#FFFFFF" LINK="#FF0000" VLINK="#505050" MARGINHEIGHT="30"
TOPMARGIN="30" LEFTMARGIN="40" MARGINWIDTH="40">
...
Текст документа.
...
</BODY>
</HTML>
```

Гиперссылки

Ссылки на другие документы в HTML создаются либо с помощью элемента A, либо с помощью навигационных карт.

Элемент A применяется, если ссылкой планируется сделать часть текста

или целое изображение. Навигационные карты имеет смысл применять, если ссылкой будет часть изображения.

A

Самый необходимый элемент, без которого Интернет просто немыслим. Используется для создания и использования гипертекстовых ссылок.

Атрибуты:

HREF – определяет находящийся между начальным и конечным тегами текст или изображение как гипертекстовую ссылку (URL, или линк) на документ (и/или область документа), указанный в значении данного атрибута. Возможные значения:

http://... – создает ссылку на www-документ;

ftp://... – создает ссылку на ftp-сайт или расположенный на нем файл;

mailto:... – запускает почтовую программу-клиент с заполненным полем имени получателя. Если после адреса поставить знак вопроса, то можно указать дополнительные атрибуты, разделенные знаком "&";

news:.. – создает ссылку на конференцию сервера новостей;

telnet://... – создает ссылку на telnet-сессию с удаленной машиной;

wais://... – создает ссылку на WAIS – сервер;

gopher://... – создает ссылку на Gopher – сервер;

Если тип соединения и адрес машины не указаны, в качестве отправной точки используется адрес текущего документа. Это позволяет использовать относительные ссылки.

Например, линк <A HREF="docs/title.html">Документация</A> будет ссылаться на файл title.html в подкаталоге docs (относительно текущего).

NAME – помечает находящуюся между начальным и конечным тегами область документа как возможный объект для ссылки. В качестве значения нужно латиницей написать любое слово-указатель, уникальное для данного документа.

Например: <A NAME="part">Раздел1</A>. Теперь вы можете ссылаться на помеченную область простым указанием ее имени после имени документа. Например, линк <A HREF="document.html#part">Раздел1</A> отправит вас в раздел "part" файла document.html, а линк <A HREF="#bottom">В конец документа</A> – в раздел "bottom" текущего документа. (см. Пример 1)

TARGET – определяет окно (фрейм), на которое указывает гипертекстовая ссылка. Этот атрибут используется только совместно с атрибутом HREF. В качестве значения необходимо задать либо имя одного из существующих фреймов, либо одно из следующих зарезевированных имен:

_self – указывает, что определенный в атрибуте HREF документ должен отображаться в текущем фрейме;

_parent – указывает, что документ должен отображаться во фрейме-родителе текущего фрейма. Иначе говоря, _parent ссылается на окно, содержащее FRAMESET, включающий текущий фрейм;

_top – указывает, что документ должен отображаться в окне-родителе всей текущей фреймовой структуры;

_blank – указывает, что документ должен отображаться в новом окне.

Пример 1:

```
<!-- Использование атрибута NAME: -->
<A NAME="history">История бодибилдинга</A>
...
<A NAME="now">Спорт глазами современника</A>
...
Вернуться к разделу<A HREF="#history">истории</A>
```

Пример 2:

```
<!-- Использование атрибута HREF: -->
<A HREF="ftp://ftp.cdrom.com" TARGET="_blank">FTP-site</A>
<A HREF="http://opengl.rdc.ru">Русский проект по OpenGL</A>
```

...

Пример 3:

<!-- Создадим ссылку для письма с указанием кучи атрибутов -->
<A HREF="mailto:green@igf.ru?subject=Приглашение
&cc=bg@microsoft.com&body=Приезжай на вечеринку.">
Отправить приглашение </A>.
<!-- или просто письмо : -->
<A HREF="mailto:green@igf.ru?subject=Привет">авторам</A>.

Текстовые блоки

В этом разделе описаны элементы, разбивающие текст документа на блоки тем или иным способом. Типичными примерами текстовых блоков являются параграфы, абзацы и главы. Для отделения одной части текста от другой также используются разделительные горизонтальные линии и символы возврата каретки.

Элементы:

H1,H2,...H6 Используются для создания заголовков текста
P Используется для разметки параграфов.
DIV Отделяет блок HTML-документа от остальной его части
ADDRESS Оформляет текст как почтовый адрес
BLOCKQUOTE Оформляет текст как цитату
BR Осуществляет перевод строки
HR Вставляет в текст горизонтальную разделительную линию.
PRE Включает в документ (моноширинным шрифтом) блок предварительно отформатированного текста

H1,H2,...H6

Используются для создания заголовков текста. Существует шесть уровней заголовков, различающихся величиной шрифта. С их помощью можно разбивать текст на смысловые уровни – разделы и подразделы.

Атрибуты:

ALIGN – определяет способ выравнивания заголовка по горизонтали. Возможные значения: left, right, center.

Пример:

<H1 ALIGN="center">Самый большой заголовок посередине</H1>
<H2>Заголовок поменьше</H2>
...
<H6>Малююююсенький такой заголовочек</H6>

P

Используется для разметки параграфов.

Атрибуты:

ALIGN – определяет способ горизонтального выравнивания параграфа. Возможные значения: left, center, right. По умолчанию имеет значение left.

Пример:

<P ALIGN="center">Это центрированный параграф.
Текст располагается в центре окна браузера</P>
<P ALIGN="right">А это параграф, выровненный по правому краю.</P>

DIV

Используется для логического выделения блока HTML-документа. Элемент группировки, как и элемент SPAN. В современном сайтостроении используется как удобный контейнер для объектов страницы, которым легко динамически манипулировать – перемещать, включать/выключать, создавать слои, регулировать отступы и т.п.

В браузеронезависимой вёрстке обычно используется для выравнивания блока html-кода в окне браузера.

Находящиеся между начальным и конечным тегами текст или HTML-элементы по умолчанию оформляются как отдельный параграф.

Атрибуты:

ALIGN – определяет выравнивание содержимого элемента DIV. Атрибут может принимать значения: left, right, center.

Пример:

...Текст документа...

<DIV ALIGN="center">

...Текст, таблицы, изображения. Выравнивание по центру.

</DIV>

...Текст документа...

ADDRESS

Находящийся между начальным и конечным тегами текст оформляется как почтовый адрес. Чаще всего оформление выражается в выделении строки адреса курсивом.

Пример:

Пишите по следующему адресу:

<ADDRESS>
Москва. ул. Академика Королева, 13
Мурзилке
</ADDRESS>

BLOCKQUOTE

Оформляет находящийся между начальным и конечным тегами текст как цитату. Используется для длинных цитат (в отличие от элемента CITE).

Цитируемый текст отображается отдельным абзацем с увеличенным отступом.

Пример:

Редакция журнала "Домосед" выражает благодарность
Бухаресту Магарычу Шницелю за замечательное стихотворение:
<BLOCKQUOTE>
Синели красные ромашки,
Желтели в небе облака,
Синицы в теплый край летели,
К истоку двигалась река.
...
</BLOCKQUOTE>

BR

Данный элемент осуществляет перевод строки, то есть практически аналогичен нажатию Enter в текстовом редакторе. После того, как в браузерах появилась возможность обтекания изображения текстом (см. атрибут ALIGN элемента IMG), понадобился дополнительный атрибут CLEAR. Элемент не имеет конечного тега.

Атрибуты:

CLEAR – указывает на необходимость завершения обтекания изображения текстом. Может принимать следующие значения :

- all – завершить обтекание изображения текстом.
- left – завершить обтекание текстом изображения, выровненного по левому краю.
- right – завершить обтекание текстом изображения, выровненного по правому краю.

Пример:

Первое предложение
Второе предложение на следующей строке

HR

Вставляет в текст горизонтальную разделительную линию.

Атрибуты:

WIDTH – определяет длину линии в пикселах или процентах от ширины окна браузера.

SIZE – определяет толщину линии в пикселах.

ALIGN – определяет выравнивание горизонтальной линии. Атрибут может принимать следующие значения:

- left – выравнивание по левому краю документа.
- right – выравнивание по правому краю документа.
- center – выравнивание по центру документа (используется по умолчанию).

NOSHADE – определяет способ закраски линии как сплошной. Атрибут является флагом и не требует указания значения. Без данного атрибута линия отображается объемной.

COLOR – определяет цвет линии. Задается либо RGB-значением в шест-

надцатиричной системе, либо одним из 16 базовых цветов. Атрибут работает только в Internet Explorer.

Пример:

Текст, разделенный <HR NOSHADE WIDTH="50%"> сплошной горизонтальной линией.

Форматирование текста

В этом разделе описаны элементы для оформления и смыслового выделения текста – подчеркивания, изменения шрифта, выделения курсивом, цитирования и т.д.

Элементы форматирования текста:

- BASEFONT Определяет основной шрифт, которым должен отображаться текст документа
- FONT Позволяет изменять цвет, размер и тип шрифта текста
- I Выделяет текст курсивом
- EM Используется для смыслового выделения текста (курсивом)
- B Выделяет текст жирным шрифтом
- STRONG Усиленное выделение текста (жирным)
- U Выделяет текст подчеркнутым
- S, STRIKE Выделяет текст перечеркнутым
- BIG Отображает текст увеличенным шрифтом (относительно текущего)
- SMALL Отображает текст уменьшенным шрифтом (относительно текущего)
- SUP Отображает текст со сдвигом вверх (верхний индекс)
- SUB Отображает текст со сдвигом вниз (нижний индекс)
- CODE, SAMP Оформляют текст как формулу или программный код
- TT Отображает текст моноширинным шрифтом
- KBD Выделяет текст, который предлагается набрать на клавиатуре

- VAR Используется для обозначения в тексте переменных
- CITE Оформляет текст как цитату или ссылку на источник

FONT

Позволяет изменять цвет, размер и тип шрифта текста, находящегося между начальным и конечным тегами. Вне тегов <FONT> и </FONT> используется шрифт, указанный в элементе BASEFONT.

Атрибуты:

SIZE – определяет размер шрифта. Возможные значения:

- целое число от 1 до 7;
- относительный размер с указанием знака, вычисляется путем сложения с базовым размером, определенным с помощью атрибута SIZE элемента BASEFONT.

COLOR – определяет цвет текста. Задается либо RGB-значением в шестнадцатиричной системе, либо одним из 16 базовых цветов.

FACE – определяет используемый шрифт.

Пример:

<FONT SIZE="+2" COLOR="#AA0000">Увеличенный красный шрифт</FONT>
<FONT SIZE="3" FACE="Courier New" COLOR="Magenta">Моноширинный фиолетовый текст 3 размера</FONT>

I

Текст, заключенный между начальным и конечным тегами, будет выделен курсивом.

Пример:

Текст с <I>курсивом</I>

EM

Логическое ударение. Используется для смыслового выделения текста, стоящего между начальным и конечным тегами. Результат обычно отображается курсивом. То есть элемент EM по действию практически аналогичен элементу I.

Пример:

Порой в России встречаются <EM>действительно талантливые</EM> веб-мастера. Но только не друг с другом.

B

Текст, заключенный между начальным и конечным тегами, будет выделен жирным шрифтом.

Пример:

Текст с <B>выделенным</B> словом

Списки

Списки в HTML бывают двух видов: упорядоченные (пронумерованные) и неупорядоченные (непронумерованные). Отличаются они лишь способом оформления. Перед пунктами неупорядоченных списков обычно ставятся символы-буллеты (bullets), например, точки, ромбики и т.п., в то время как пунктам упорядоченных списков предшествуют их номера.

Элементы:

- UL Создает неупорядоченный список
- OL Создает упорядоченный список

- LI Создает пункт меню внутри элементов OL или UL
- MENU, DIR Создает неупорядоченный список, подобный UL
- DL Открывает и закрывает список определений
- DT Создает термин в списке определений внутри элемента DL
- DD Создает определение термина внутри элемента DL

UL

Создает неупорядоченный список. Между начальным и конечным тегами должны присутствовать один или несколько элементов LI, обозначающих отдельные пункты списка.

Пример:

<UL>
<LI> Первый пункт списка </LI>
<LI> Второй пункт списка </LI>
<LI> Третий пункт списка </LI>
</UL>

OL

Создает упорядоченный список. Между начальным и конечным тегами должны присутствовать один или несколько элементов LI, обозначающих отдельные пункты списка.

Атрибуты:

START – определяет первое число, с которого начинается нумерация пунктов. (только целые числа)

TYPE – определяет стиль нумерации пунктов. Может иметь значения:

- "A" – заглавные буквы A, B, C ...
- "a" – строчные буквы a, b, c ...

- "I" – большие римские числа I, II, III ...
- "i" – маленькие римские числа i, ii, iii ...
- "1" – арабские числа 1, 2, 3 ...

  По умолчанию <UL TYPE="1">.

Пример:

<OL TYPE="I" START="2">
<LI> Пункт два </LI>
<LI> Пункт три </LI>
<LI> Пункт четыре </LI>
</OL>

LI

Создает пункт в списке. Располагается внутри элементов OL или UL.

Атрибуты:

VALUE – изменяет порядок нумерации элементов списка. Используется только если элемент LI находится внутри элемента OL. В качестве значения указывается порядковый номер элемента.

Пример:

<OL TYPE="A" START="2">
<LI> Пункт, озаглавленный буквой B. </LI>
<LI VALUE="6"> Пункт, озаглавленный буквой F. </LI>
<LI> Пункт, озаглавленный буквой G. </LI>
</OL>

Объекты

Объекты – это графические и мультимедийные вставки в HTML-документ, такие как картинки, Flash-анимация, Java-апплеты, звуки, музыка, VRML.

Элементы:

- IMG Используется для вставки в HTML изображений
- EMBED Используется для вставки в HTML различных объектов
- NOEMBED Используется, если браузер не поддерживает элемент EMBED
- APPLET Используется для вставки в HTML Java-апплетов
- PARAM Используется для передачи параметров Java-программе (см. элемент APPLET)

IMG

Используется для вставки изображений в HTML-документ.

Это один из самых популярных элементов, незаменимый инструмент web-дизайнера. Элемент допускает вставку изображений в форматах JPEG (в том числе progressive jpeg) и Compuserve GIF (включая прозрачные и анимированные). Червертые версии браузеров позволяют также использовать формат PNG, но до тех пор, пока они не устареют, от применения PNG лучше воздержаться.

Элемент IMG не имеет конечного тега.

Атрибуты:

SRC – обязательный атрибут. Указывает адрес (URL) файла с изображением.

HEIGHT и WIDTH – определяют ширину и высоту изображения соответственно. Если указанные значения не совпадают с реальным размером изображения, изображение масштабируется (порой с заметной потерей качества).

HSPACE и VSPACE – определяют отступ картинки (в пикселах) по горизонтали и вертикали от других объектов документа. Просто необходимо при об-

текании изображения текстом.

ALIGN – обязательный атрибут. Указывает способ выравнивания изображения в документе. Может принимать следующие значения:

- left – выравнивает изображение по левому краю документа. Прилегающий текст обтекает изображение справа.
- right – выравнивает изображение по правому краю документа. Прилегающий текст обтекает изображение слева.
- top и texttop – выравнивают верхнюю кромку изображения с верхней линией текущей текстовой строки.
- middle – выравнивает базовую линию текущей текстовой строки с центром изображения.
- absmiddle – выравнивает центр текущей текстовой строки с центром изображения.
- bottom и baseline – выравнивает нижнюю кромку изображения с базовой линией текущей текстовой строки.
- absbottom – выравнивает нижнюю кромку изображения с нижней кромкой текущей текстовой строки.

NAME – определяет имя изображения, уникальное для данного документа. Вы можете указать любое имя без пробелов с использованием латинских символов и цифр. Имя необходимо, если вы планируете осуществлять доступ к изображению, например, из JavaScript-сценариев.

ALT – определяет текст, отображаемый браузером на месте изображения, если браузер не может найти файл с изображением или включен в текстовый режим. В качестве значения задается текст с описанием изображения.

BORDER – определяет ширину рамки вокруг изображения в пикселах. Рамка возникает, только если изображение является гипертекстовой ссылкой. В таких случаях значение BORDER обычно указывают равным нулю.

LOWSRC – указывает адрес (URL) файла с альтернативным изображением более низкого качества (и, соответственно, меньшего объема), чем изображение, указанное в атрибуте SRC. Браузеры Netscape, поддерживающие данный

атрибут, сначала загрузят картинку из LOWSRC, а затем заменят ее картинкой из SRC.

USEMAP – применяет к изображению навигационную карту (image map), заданную элементом MAP. В качестве значения задается имя карты с предшествующей ему решеткой. Например, если имя карты – "map1", то ссылка на нее будет выглядеть как "#map1" (см. Пример 4). Обратите внимание: прописные и строчные буквы в данном атрибуте трактуются браузером как разные.

ISMAP – определяет изображение как навигационную карту (image map), обрабатываемую сервером. Имеет смысл использовать только тогда, когда изображение является гиперссылкой. После клика мышкой на изображении серверу отправляются x,y-координаты нажатия. В зависимости от полученных координат, сервер (при наличии на нем соответствующего программного обеспечения) может показать вам тот или иной документ. Данный атрибут является флагом и не требует присвоения значения.

Пример 1:

```
<IMG src="/img/picture.gif" WIDTH="45" HEIGHT="53" ALT="Рысь"
HSPACE="10" ALIGN="left"> Этот текст обтекает картинку справа и находится от нее на расстоянии 10 пикселов.
```

Пример 2. Использование изображения в качестве гиперссылки:

```
<A HREF="carlson.html">
<IMG src="/img/button.jpg" WIDTH="70" HEIGHT="30" ALIGN="right"
BORDER="0" ALT="Досье Карлсона">
</A>
Для просмотра досье нажмите на кнопку справа.
```

Пример 3. Использование ISMAP:

```
<A HREF="http://www.igf.ru/bin/imagemaps/map1">
<IMG SRC="map.gif" ISMAP></A>');
```

Пример 4. Использование USEMAP:

<IMG src="/img/buttons.jpg" WIDTH="170" HEIGHT="120" ALIGN="middle" BORDER="0" USEMAP="#ButtonsMap">');

Примечания (особо важно):

Золотое правило web-мастера – всегда явно задавать размеры картинки в атрибутах HEIGHT и WIDTH, резервируя тем самым место в окне браузера еще до загрузки изображения. В противном случае документ при загрузке каждой картинки будет заново перерисовываться. А на медленных машинах (или при подключении к сети через модем) это смотрится просто отвратительно.

Второе золотое правило web-мастера: если на картинке изображено что-то разборчивое, нужно описать это словами в атрибуте ALT.

Всегда сразу после <IMG ...> ставьте
! А то проблем не миновать – после картинки появится пустое пространство в несколько пикселей. Причём ставьте вплотную, без пробелов: <img ...>
.

Для завершения обтекания изображения текстом используйте атрибут CLEAR элемента BR.

Значения top и texttop атрибута ALIGN не совсем идентичны, и их использование порой дает разный результат. Попробуйте поэкспериментировать.

Указывайте значения атрибутов HSPACE и VSPACE, даже если вы хотите оставить поля нулевой ширины. Бывает, что некоторые браузеры по умолчанию присваивают им какое-то небольшое значение, не равное нулю.

Таблицы

Элементы для создания таблиц:

Таблицы в HTML формируются нетрадиционным способом – построчно. Сначала с помощью элемента TR необходимо создать ряд таблицы, в который затем элементом TD помещаются ячейки.

Важно:

В HTML таблицы используются не только для отображения таблиц как таковых, но и для дизайна. С помощью таблиц можно создать невидимый "каркас" страницы, помогающий расположить текст и изображения так, как вам нравится.

Элементы для создания таблиц:

- TABLE Создает таблицу
- CAPTION Задает заголовок таблицы
- TR Создает новый ряд (строку) ячеек таблицы
- TD и TH Создает ячейку с данными в текущей строке

TABLE

Элемент для создания таблицы. Обязательно должен иметь начальный и конечный теги. По умолчанию таблица печатается без рамки, а разметка осуществляется автоматически в зависимости от объема содержащейся в ней информации. Ячейки внутри таблицы создаются с помощью элементов TR, TD, TH и CAPTION.

Атрибуты:

ALIGN – определяет способ горизонтального выравнивания таблицы. Возможные значения: left, center, right. Значение по умолчанию – left.

VALIGN – должен определять способ вертикального выравнивания таблицы. Возможные значения: top, bottom, middle.

BORDER – определяет ширину внешней рамки таблицы (в пикселах). При BORDER="0" или при отсутствии этого атрибута рамка отображаться не будет.

CELLPADDING – определяет расстояние (в пикселах) между рамкой каждой ячейки таблицы и содержащимся в ней материалом.

CELLSPACING – определяет расстояние (в пикселах) между границами соседних ячеек.

WIDTH – определяет ширину таблицы. Ширина задается либо в пикселах, либо в процентном отношении к ширине окна браузера. По умолчанию этот атрибут определяется автоматически в зависимости от объема содержащегося в таблице материала.

HEIGHT – определяет высоту таблицы. Высота задается либо в пикселах, либо в процентном отношении к высоте окна браузера. По умолчанию этот атрибут определяется автоматически в зависимости от объема содержащегося в таблице материала.

BGCOLOR – определяет цвет фона ячеек таблицы. Задается либо RGB-значением в шестнадцатиричной системе, либо одним из 16 базовых цветов.

BACKGROUND – позволяет заполнить фон таблицы рисунком. В качестве значения необходимо указать URL рисунка.

TR

Создает новый ряд (строку) ячеек таблицы. Ячейки в ряду создаются с помощью элементов TD и TH

Атрибуты:

ALIGN – определяет способ горизонтального выравнивания содержимого всех ячеек данного ряда. Возможные значения: left, center, right.

VALIGN – определяет способ вертикального выравнивания содержимого всех ячеек данного ряда. Возможные значения: top, bottom, middle.

BGCOLOR – определяет цвет фона для всех ячеек данного ряда. Задается либо RGB-значением в шестнадцатиричной системе, либо одним из 16 базовых цветов.

TD и TH

Элемент TD создает ячейку с данными в текущей строке. Элемент TH также создает ячейку, но определяет ее как ячейку-заголовок.

Такое разграничение позволяет браузерам оформлять содержимое ячейки-заголовка и ячеек с данными разными шрифтами. Кроме того, должна улучшиться работа браузеров, использующих речевой интерфейс. В качестве содержимого ячейки можно использовать другие таблицы.

Атрибуты:

ALIGN – определяет способ горизонтального выравнивания содержимого ячейки. Возможные значения: left, center, right. По умолчанию способ выравнивания определяется значением атрибута ALIGN элемента TR. Если же и он не задан, то для TD выполняется выравнивание по левому краю, а для TH – центрирование.

VALIGN – определяет способ вертикального выравнивания содержимого ячейки. Возможные значения: top, bottom, middle. По умолчанию происходит выравнивание по центру (VALIGN="middle"), если значение этого атрибута не было задано ранее в элементе TR.

WIDTH – определяет ширину ячейки. Ширина задается в пикселах или в процентном отношении к ширине таблицы.

HEIGHT – определяет высоту ячейки. Высота задается в пикселах или в процентном отношении к высоте таблицы.

COLSPAN – определяет количество столбцов, на которые простирается данная ячейка. По умолчанию имеет значение 1.

ROWSPAN – определяет количество рядов, на которые простирается данная ячейка. По умолчанию имеет значение 1.

NOWRAP – блокирует автоматический перенос слов в пределах текущей ячейки. Обратите внимания на примечание, касающееся использования данного атрибута (далее, внизу страницы).

BGCOLOR – определяет цвет фона ячейки. Задается либо RGB-значением

в шестнадцатиричной системе, либо одним из 16 базовых цветов.

BACKGROUND – заполняет ячейку фоновым рисунком. Необходимо указать URL рисунка. Данный атрибут не работает в старых версиях браузера Netscape (до 3.X включительно).

Лабораторная работа №1. Создание мини-сайта.

Цель работы

Знакомство с гипертекстовым языком разметки HTML. Формирование структуры мини-сайта. Верстка основных страниц будущего приложения информационной системы.

Задание на лабораторную работу

Создать структуру каталогов будущего мини-сайта в соответствии с описанной выше.

Создать в каждом каталоге индексные файлы index.phtml.

Выполнить оформление и осуществить верстку web-страниц.

Порядок выполнения работы

Создание мини-сайта производится каждым студентом индивидуально. Страницы располагаются на web-сервере кафедры электронных приборов в каталоге stdweb. Доступ к серверу осуществляется по протоколу ftp. Логин для входа на сервер и пароль предоставлются преподавателем.

Для упорядочения файлов и каталогов на сервере свои файлы студенты должны размещать в слудующем порядке:

/stdweb/<номер группы>/<логин студента>

Например:

/stdweb/353-2/3532-yas/

В операционной системе Linux в окружении KDE доступ к серверу по протоколу ftp в интерактивном режиме может быть осуществлен в частности следующими способами:

- с помощью файлового менеджера Midnight Commander (mc);

- с помощью файлового менеджера Konqueror;
- непосредственно с помощью программы текстового реактора Kwrite.

Мы рекомендуем использовать последний способ поскольку он позволяет одновременно и редактировать документы и открывать/сохранять их на сервере.

Обращаем особое внимание на то что кодировкой по умолчанию в Linux является UTF-8. В российском сегменте интернет пока общепризнанной кодировкой является Windows-1251 (или CP-1251). Переход на Unicode кодировки только начинает происходить. В этой связи настоятельно рекомендуется студентам создавать свои документы в кодировке CP-1251. Выбрать кодировку можно с помощью меню редактора Kwrite Tools – Encoding – Cyrillic (cp-1251). В русифицированном варианте названия пунктов могут быть другими. Программа Kwrite, если ее нет в пользовательском меню, может быть вызвана непосредственно: /usr/bin/kwrite.

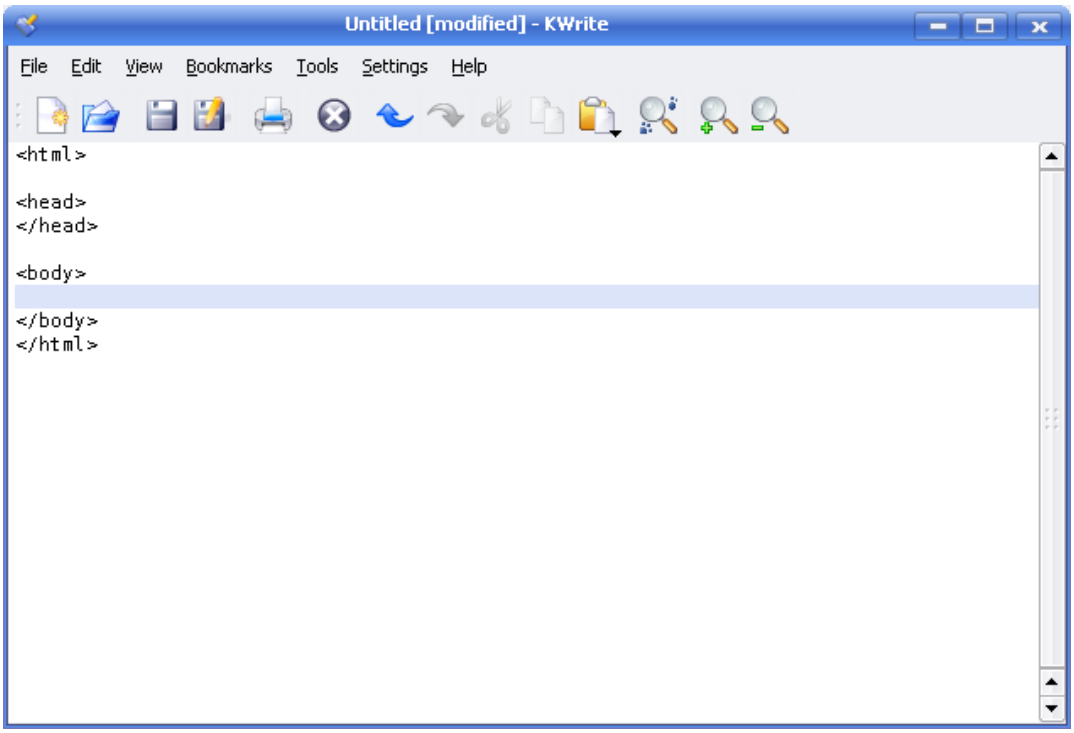

Рис. 1. Внешний вид программы Kwrite.

Для доступа к файлам на сервере по протоколу ftp при открытии файла или при его создании используйте следующий путь:

ftp://<логин ftp>@ed.tusur.ru

Например:

ftp://stdweb@ed.tusur.ru

Программа попросит затем вас ввести пароль на доступ к ftp-серверу. После этого можно работать с сервером так же как это происходит при работе с локальными файлами.

В выделенном на web-сервере каталоге создайте каталог с номером вашей группы (если он уже не создан), затем каталог имя которого соответсвует вашему логину (например 3522-yas). В этой папке затем происходит вся работа.

Создайте следующую структуру каталогов и файлов в вашей папке (от корня вашей папки):

index.phtml

about

    index.phtml

admin

    index.phtml

catalog

    index.phtml

Здесь каталог about предназначен для размещения информации о вашем сайте, catalog используется для доступа к информации о каталоге товаров, представленных в вашем магазине, папка admin используется в целях администрирования (управления) магазином. Файл index.phtml является индексным, то есть при указании броузеру адреса

http://ed.tusur.ru/stdweb/352-2/3522-yas/catalog

будет показано содержимое файла index.phtml находящегося в папке catalog. В этом случае указывать имя файла не обязательно. Во всех остальных случаях (когда имя файла отличается от index) необходимо в явном виде указывать имя вызываемого файла.

Выбор макета страниц (варианта верстки) студенты могут сделать самостоятельно. В данном пособии мы приведем лишь некоторые рекомендации, ко-

торые могут помочь при выполнении этого задания.

Постараемся выделить важные разделы макета и основные принципы его формирования.

Выбор варианта верстки

Верстка страницы может быть произведена двумя способами:

- использовать фиксированную ширину страницы;
- использовать всю (100%) ширину страницы.

Правильного выбора в данном случае не существует. И тот и другой вариант равнозначны и имеют как свои плюсы так и минусы. Тем не менее, мы рекомендуем использовать второй вариант, как наиболее предпочтительный в настоящий момент.

Система навигации по сайту

Поскольку наш мини-магазин, а точнее его витрина, представляют собой список товаров, распределенный по категориям рекомендуется использовать т.н. “вертикальный” способ навигации для выбора категории товара, а “горизонтальный” для перемещения по разделам сайта.

Макет

Заголовочная часть. Располагается в верхней части страницы. Занимает 100% ширины. Высота от 100 до 200 пикселов. Должна включать в себя либо графический логотип, либо текстовое название сайта. Кроме этого здесь же располагается “горизонтальная” система меню, предназначенная для перемещения по разделам сайта.

Основная часть. Делится на две: правая для размещения меню раздела (например категорий), основная (центральная) предназначена для размещения информации. Для правой части предлагается выбрать 20% ширины, для основной – оставшиеся 80%.

“Подвал”. Предназначен для размещения контактной информации. Занимает 100% ширины. Здесь необходимо указать контакты нашего магазина, например телефон и адрес электронной почты. Расположить эти элементы целесообразно с выравниванием по левому краю.

Верстка

Несмотря на то что язык HTML позволяет осуществлять выбор как шрифта, так и размера и цвета для текста, мы не рекомендуем пользоваться этим приемом. Для шрифтового оформления наших документов лучше использовать каскадные таблицы стилей, которые мы введем в наш мини-сайт позже.

Как уже было сказано выше, макет нашего сайта должен занимать 100% ширины экрана. В этом случае, когда основная часть располагается слева, как правило затруднено чтение текста прижатого к левой границе окна броузера. Это можно исправить несколькими способами, предпочтительным является использование каскадных таблиц стилей.

Планирование приложения

Несмотря на то что мы пока не работаем с реальными данными, студентам необходимо выбрать тематику их будущего магазина. Никаких особых требований здесь нет. Рекомендации следующие:

- число товаров от 20 до 50;
- число категорий товаров 5 – 7.

Отчет

По окончании работы студенты готовят отчет и предоставляют его преподавателю в электронном виде в формате PDF. Отчет должен разделы: цель работы, порядок выполнения, основная часть должна содержать скриншоты страниц мини-сайта в броузере, текст HTML-кода страниц. Отчет должен заканчиваться выводом.

Язык Web-программирования PHP

Что такое PHP

PHP (рекурсивный акроним для "PHP: Hypertext Preprocessor") это широко распространённый язык скриптинга (сценариев) общего назначения, который создан специально для Web и который можно внедрять в HTML.

Пример использования языка PHP

```
<html>
<head>
<title>Пример</title>
</head>
<body>

<?php
    echo "Hello World!";
?>

</body>
</html>
```

Заметьте, как это отличается от скриптов, написанных на языках Perl или C - вместо написания программы с большим количеством команд для вывода HTML, вы пишете HTML-скрипт с некоторым количеством встроенного кода для выполнения каких-либо действий (в данном случае - для вывода некоторого текста). Код PHP заключён в специальные начальный и конечный тэги, что позволяет вам входить в и выходить из "режима PHP".

PHP отличается от других подобных языков, типа клиентского JavaScript,

тем, что код выполняется на сервере. Если вы имеете скрипт, аналогичный вышеприведённому на сервере, то клиент получит результат работы этого скрипта, не имея возможности определить, каков был исходный код.

Наилучшим качеством PHP является то, что он предельно прост для новичка в программировании, но предлагает много продвинутых возможностей для программиста-профессионала.

Что может PHP?

PHP в основном сориентирован на серверный скриптинг, поэтому может делать всё то, что делают CGI-программы: сбор данных форм, динамическую генерацию содержимого страницы или приём и отправку кук. Но PHP может намного больше.

Скрипты PHP применяются в трёх основных сферах.

- Серверный скриптинг. Это наиболее традиционная и главная сфера применения PHP. Для выполнения этой работы вам нужны три вещи. Разборщик кода PHP (CGI или серверный модуль), web-сервер и web-браузер. Сервер должен быть запущен и должен иметь соединение с инсталированным PHP. Вы можете получить вывод PHP-программы в web-браузер, просматривая PHP-страницу на сервере.
- Скриптинг командной строки. Вы можете создать и запустить PHP-скрипт на выполнение без сервера или браузера. Для этого необходим только разборщик PHP. Этот тип использования идеально подходит для регулярного выполнения скрипта с помощью cron (в *nix или Linux) или Task Scheduler (в Windows). Эти скрипты можно использовать также для задач простейшего текстового процессинга/обработки.
- Клиентские GUI-приложения. PHP, возможно, не самый лучший язык для написания оконных приложений, но, если вы знаете PHP очень хорошо и хотели бы использовать некоторые продвинутые возможности PHP в клиентских приложениях, вы можете также использовать PHP-GTK для создания таких программ. У вас имеется также возможность создавать меж-

платформенные приложения. PHP-GTK является расширением PHP, отсутствующим в основном дистрибутиве.

PHP может использоваться на всех крупных операционных системах (ОС), включая Linux, многие варианты Unix (HP-UX, Solaris и OpenBSD), Microsoft Windows, Mac OS X, RISC OS и, возможно, другие. PHP имеет поддержку для большинства существующих web-серверов. Это Apache, Microsoft Internet Information Server, Personal Web Server, Netscape и iPlanet-серверы, Oreilly Website Pro, Caudium, Xitami, OmniHTTPd и многие другие. Для большинства этих серверов PHP имеет модули. В других, поддерживающих стандарт CGI, PHP может работать как CGI-процессор.

Итак, с помощью PHP вы получаете свободу выбора ОС и web-сервера. Более того, вы может также выбрать использование процедурного или объектно-ориентированного варианта программирования или их сочетания. Хотя не всякая стандартная возможность ООП реализована в текущей версии PHP, многие библиотеки кодов и большие приложения (включая библиотеку PEAR) написаны только с использованием ООП-кода.

В PHP вы не имеете ограничений в выводе HTML. PHP может выводить изображения, PDF-файлы и даже клипы Flash (используя libswf и Ming), генерируемые на лету. Вы также легко можете выводить любой текст, включая XHTML, и любой другой XML-файл. PHP может автоматически генерировать эти файлы и сохранять их в файловой системе, вместо их распечатки, формируя серверный кэш для вашего динамического содержимого.

Одна из наиболее сильных и привлекательных черт PHP - поддержка им большого количества баз данных (БД). Создать web-страницу, работающую с БД, невероятно легко.

PHP поддерживает взаимодействие с другими службами по таким протоколам, как LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (под Windows) и множество других. Вы можете также открыть обычный сетевой сокет и взаимодействовать с использованием любого другого протокола.

PHP имеет предельно удобные возможности для работы с текстом, от

POSIX Extended или регулярных выражений Perl до разбора документов XML.

Внедрение в HTML

Когда PHP разбирает файл, он просто передаёт текст файла, пока не обнаружит один из специальных тэгов, который говорит о необходимости начать интерпретацию текста как кода PHP. Разборщик выполняет весь найденный код до закрывающего тэга PHP, который говорит разборщику, что нужно снова начать просто передавать текст. Этот механизм позволяет внедрять PHP-код в HTML: всё за пределами тэгов PHP остаётся без изменений, а внутри тэгов - разбирается как код.

Имеются четыре набора тэгов, которые используются для обозначения блоков кода PHP.

Только два из них (<?php. . .?> и <script language="php">. . .</script>) всегда доступны; другие можно включать и отключать из файла конфигурации php.ini. Хотя сокращённые тэги и тэги в стиле ASP могут быть удобны, они не так переносимы, как их длинные версии. Также, если вы предполагаете внедрять PHP-код в XML или XHTML, нужно использовать форму <?php. . .?> для соответствия XML.

Тэги, поддерживаемые PHP:

1. <?php echo("если вы хотите работать с документами XHTML или XML, делайте так\n"); ?>
2. <? echo ("это простейшая SGML-инструкция процессинга\n"); ?>
   <?= выражение ?> Это аббревиатура для "<? echo выражение ?>"
3. <script language="php">
echo ("некоторые редактроы (вроде FrontPage) не любят
инструкции процессинга");
</script>
4. <% echo ("Вы можете по выбору использовать тэги в стиле ASP"); %>
 <%= $variable; # Это аббревиатура для "<% echo . . ." %>

Первый способ, <?php. . .?>, это предпочтительный метод, так как он позволяет использовать PHP в коде, соответствующем правилам XML, таком как XHTML.

Типы данных в PHP

PHP поддерживает 8 примитивных типов.

4 скалярных типа:

- boolean
- integer
- число с плавающей точкой (float)
- string

Два составных типа:

- array
- object

И, наконец два специальных типа:

- resource
- NULL

В дальнейшем изложении вы часто будете встречать mixed-параметры. Этот псевдотип указывает обозначает несколько возможностей для данного параметра.

Тип переменной обычно программистом не устанавливается; напротив, он определяется PHP на этапе прогона, в зависимости от контекста, в котором эта переменная используется.

Переменные

Переменные в PHP представлены знаком dollar ($) с последующим именем переменной. Имя переменной чувствительно к регистру символов.

Имена переменных следуют тем же правилам, что и другие метки в PHP. Правильное имя переменной начинается с буквы или символа подчёркивания, с последующими (в любом количестве) буквами, числами или символами подчёркивания. Это можно выразить в виде регулярного выражения: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";      // выводит "Bob, Joe"

$4site = 'not yet';     // неправильно; начинается с числа
$_4site = 'not yet';    // правильно; начинается с символа подчёркивания/underscore
```

Предопределённые переменные

В PHP имеется большое количество предопределённых переменных, доступных любому скрипту. Многие эти переменные, однако невозможно полностью задокументировать, так как они зависят от сервера, на котором происходит работа, его версии и установок и других факторов. Некоторые из этих переменных будут недоступны при запуске PHP из командной строки.

Начиная с версии 4.1.0, PHP предоставляет набор предопределённых массивов, содержащих переменные web-сервера (если они имеются), окружения и пользовательского ввода. Эти новые массивы более специализированы, так как автоматически являются глобальными - т.е. автоматически доступны в любой области видимости.

Суперглобалы PHP

$GLOBALS - Содержит ссылку на каждую переменную, доступную в данный момент в глобальной области видимости данного скрипта. Ключами этого массива являются имена глобальных переменных.

$_SERVER - Переменные, установленные web-сервером или как-либо иначе относящиеся к среде окружения выполнения текущего скрипта.

$_GET - Переменные, предоставляемые скрипту через HTTP GET.

$_POST - Переменные, предоставляемые скрипту через HTTP POST.

$_COOKIE - Переменные, предоставляемые скрипту через HTTP cookies.

$_FILES - Переменные, предоставляемые скрипту через HTTP post-загрузку файлов.

$_ENV - Переменные, предоставляемые скрипту через среду окружения.

$_REQUEST - Переменные, предоставляемые скрипту через любой механизм пользовательского ввода, и которым, следовательно, нельзя доверять/trust.

Серверные переменные: $_SERVER

$_SERVER это массив, содержащий такую информацию, как headers/шапки, paths/пути и размещение скриптов. Вхождения в этом массиве создаются web-сервером. Нет гарантии, что каждый web-сервер будет предоставлять что-нибудь из этого; сервер может опустить позиции, указанные здесь, или добавить новые, здесь не указанные. Насчитывается большое количество этих переменных для спецификации CGI 1.1, поэтому вы должны это учитывать.

Это 'суперглобальная', или автоматическая, переменная. Это просто означает, что она доступна во всех областях видимости в скрипте.

Вы можете или можете не найти следующие переменные элементы в $_SERVER. Обратите внимание, что лишь некоторые (если вообще какие-нибудь) из этих элементов будут доступны (или будут иметь иное значение) при запуске PHP из командной строки.

'PHP_SELF' - Имя файла исполняемого в данный момент скрипта; относительно document root. Например, $_SERVER['PHP_SELF'] в скрипте с адресом http://example.com/test.php/foo.bar даст /test.php/foo.bar. Если PHP запущен как процессор командной строки, эта переменная недоступна.

'argv' - Массив аргументов, передаваемых скрипту. Если скрипт работает из командной строки, это даёт доступ, в стиле C, к параметрам командной строки. Если вызывается через метод GET, будет содержать строку запроса.

'argc' - Содержит количество параметров командной строки, передаваемых скрипту (если запущен из командной строки).

'GATEWAY_INTERFACE' - Какой вариант спецификации CGI используется сервером; например, 'CGI/1.1'.

'SERVER_NAME' - Имя хоста сервера, на котором текущий скрипт выполняется. Если скрипт запущен на виртуальном хосте, это будет значение, определённое для данного виртуального хоста.

'SERVER_SOFTWARE' - Строка-идентификатор сервера, даваемая в шапках/headers при ответах на запросы.

'SERVER_PROTOCOL' - Имя и версия информационного протокола, по которому страница запрошена; например, 'HTTP/1.0';

'REQUEST_METHOD' - Какой метод запроса был для доступа к странице; например, 'GET', 'HEAD', 'POST', 'PUT'.

'QUERY_STRING' - Строка запроса, если имеется, по которому был выполнен доступ к странице.

'DOCUMENT_ROOT' - Корневая директория документов, под которой выполняется текущий скрипт, как определено в файле конфигурации сервера.

'HTTP_REFERER' - Адрес страницы (если имеется), который направил пользовательского агента (ПА) на текущую страницу. Устанавливается ПАгентом. Не все ПА будут его устанавливать, а некоторые могут модифицировать HTTP_REFERER. Короче говоря, доверять ему нельзя.

'REMOTE_ADDR' - IP-адрес, с которого пользователь просматривает текущую страницу.

'PHP_AUTH_USER' - При работу под Apache-модулем и выполнении HTTP-аутентификации, в эту переменную устанавливается username, предоставляемое пользователем.

'PHP_AUTH_PW' - При работу под Apache-модулем и выполнении HTTP-аутен-

тификации, в эту переменную устанавливается password, предоставляемый пользователем.

'PHP_AUTH_TYPE' - При работу под Apache-модулем и выполнении HTTP-аутентификации, в эту переменную устанавливается тип аутентификации.

Выражения

Выражения это краеугольный камень PHP. В PHP вы почти всё записываете в виде выражений. Точнее и проще всего определить выражение как "нечто, имеющее значение".

Базовыми формами выражений являются константы и переменные. Если вы записываете "$a = 5", вы присваиваете '5' переменной $a. '5', очевидно, имеет значение 5 или, другими словами, '5' это выражение со значением 5 (в данном случае '5' это целочисленная/integer константа).

После этого присвоения вы предполагаете, что $a имеет значение 5, поэтому, если вы записываете $b = $a, вы ожидаете, что результат будет таким же, что и при записи $b = 5. Иначе говоря, $a это выражение со значением 5. Если всё работает правильно, результат будет именно таким.

Другой хороший пример ориентации на выражения - пре- и пост-инкремент и декремент. Пользователи PHP/FI 2 и многих других языков уже наверняка знакомы с нотацией переменная++ и переменная--. Это операции инкремента и декремента. В PHP/FI 2 оператор '$a++' не имеет значения (не является выражением), и, таким образом, вы не можете присвоить его или использовать иным образом. PHP улучшает возможности операций increment/decrement, делая эти выражения также выражениями, как в C. В PHP, как и в C, есть два типа инкремента: pre-increment и post-increment. И pre-increment, и post-increment увеличивают значение переменной на 1, и значения переменной идентичны. Разница в значении выражения инкремента. Pre-increment, который записывается как '++ $variable', вычисляется в новое значение (PHP сначала увеличивает значение, прежде чем его прочитать, отсюда название 'pre-increment'). Post-increment, ко-

торый записывается '$variable++', вычисляет оригинальное значение переменной $variable, а затем выполняет инкремент (PHP увеличивает переменную после чтения её значения, отсюда название 'post-increment').

Очень распространённый тип выражений - выражения сравнения. Они вычисляются в 0 или 1, означая FALSE или TRUE (соответственно). PHP поддерживает > (больше), >= (больше или равно), == (равно), != (не равно), < (меньше) и <= (меньше или равно). Эти выражения чаще всего используются внутри условных операторов, таких как if.

Последний пример выражений, рассматриваемых нами здесь, это комбинированные выражения вида операция-присвоение. Вы уже знаете, что, если вы хотите увеличить $a на 1, вы просто записываете '$a++' или '++$a'. Но что, если вы хотите добавить больше, чем 1, например, 3? Вы можете, конечно, записать '$a++' несколько раз, но это, очевидно, не самый эффективный и удобный способ. Чаще всего записывают '$a = $a + 3'. '$a + 3' вычисляется в значение $a плюс 3, и оно присваивается обратно переменной $a, что увеличивает значение $a на 3. В PHP, как и в некоторых других языках вроде C, вы можете записать это сокращённо, что со временем станет легче читать и понимать. Добавить 3 к текущему значению $a можно, записав '$a += 3'. Это означает "взять значение переменной $a, прибавить к нему 3 и присвоить новое значение этой же переменной $a". Помимо того, что это понятнее, это также ускоряет выполнение. Значением '$a += 3', как значением регулярного выражения, является присвоенное значение. Заметьте, что это НЕ 3, а объединённое значение переменной $a плюс 3. Любая двухместная операция может использоваться в этом режиме операция-присвоение, например, '$a -= 5' (вычесть 5 из значения переменной $a), '$b *= 7' (умножить значение переменной $b на 7), etc.

Структуры Управления

Любой PHP-скрипт состоит из серии операторов. Это может быть присвоение, вызов функции, цикл, условный оператор или даже оператор, который ничего не делает (пустой оператор).

Оператор обычно завершается точкой с запятой. Кроме того, операторы можно группировать с помощью фигурных скобок {}. Группа операторов сама также является оператором.

if

Конструкция if является одной из ключевых во многих языках, в том числе и в PHP. Она позволяет выполнять фрагменты кода при выполнении условия.

if (expr) statement

Как сказано в разделе о выражениях, expr вычисляется в булево значение. Если expr вычисляется в TRUE, PHP выполнит statement, а если вычисляется в FALSE - оператор игнорируется.

else

Часто нужно выполнить оператор, если соблюдено какое-либо условие, и другой оператор - если условие не соблюдено. Для этого предназначен оператор else.

else расширяет оператор if и выполняет свои операторы, если проверяемое выражение в операторе if вычисляется в FALSE. Например, следующий код выведет a is bigger than b, если $a окажется больше $b, и a is NOT bigger than b - в противном случае:

```
if ($a > $b) {
        print "a is bigger than b";
} else {
        print "a is NOT bigger than b";
}
```

Оператор else выполняется только в том случае, если выражение if вычисляется в FALSE.

while

Циклы while это простейшие циклы PHP. Вот базовая форма оператора while:

while (expr) statement

Значение оператора while говорит PHP, что нужно неоднократно выполнять вложенный оператор (-ы), пока выражение while вычисляется в TRUE. Значение выражения проверяется каждый раз в начале цикла, поэтому, если это значение изменилось при выполнении вложенного оператора (-ов), выполнение не остановиться до конца данной итерации (каждый раз, когда PHP выполняет все операторы цикла, называется одной итерацией цикла). Иногда, если выражение while вычисляется в FALSE в самом начале цикла, вложенный оператор (-ы) может быть не выполнен ни разу.

```
$i = 1;
while ($i <= 10) {
        print $i++;
}
```

for

Циклы for это самые сложные циклы PHP. Синтаксис цикла for таков:

for (expr1; expr2; expr3) statement

Первое выражение (expr1) вычисляется (выполняется) один раз и безусловно в начале выполнения цикла.

При начале каждой итерации вычисляется expr2. Если оно вычисляется в TRUE, цикл продолжается и выполняется вложенный (-ые) оператор (-ы). Если оно вычисляется в FALSE, выполнение цикла прекращается.

В конце каждой итерации вычисляется (выполняется) expr3.

Пример: выводятся числа от 1 до 10:

```
for ($i = 1; $i <= 10; $i++) {
        print $i;
}
```

Что еще?

Разумеется, возможности PHP не ограничиваются вышеизложенными. Но, поскольку список возможностей и справочник функций слишком велики, мы предлагаем студентам воспользоваться ими самостоятельно. Справочную информацию можно найти на сайте кафедры ЭП в разделе Лаборатории информационных технологий: http://ed.tusur.ru/lit.

Лабораторная работа №2. Создание каталога товаров.

Цель работы

Первичное знакомство с языком web-программирования PHP. Проектирование и планирование будущей информационной системы. Подготовка структур данных и первичное наполнение БД. Вывод списка товаров и информации по каждому продукту.

Задание на лабораторную работу

Спланировать будущую информационную систему. Спроектировать структуры данных для БД интернет-магазина.

Создать обновленную структуру каталогов информационной системы интернет-магазина. Создать и произвести первичное заполнение данными БД магазина.

Создать скрипты, осуществляющие вывод информации о списке товаров в каталоге и подробной информации по каждому товару.

Порядок выполнения работы

Выполнение лабораторной работы производится каждым студентом индивидуально. PHP-скрипты располагаются на web-сервере кафедры электронных приборов в каталоге stdweb. Доступ к серверу осуществляется по протоколу ftp. Логин для входа на сервер и пароль предоставлются преподавателем.

Создание системы каталогов

Прежде всего необходимо создать дополнительные каталоги для работы нашей информационной системы. В корне своего мини-сайта создайте папку data. Здесь мы будем хранить все данные, которые будут использоваться систе-

мой. Внутри папки data создайте каталог photos. Здесь будут храниться файлы с изображениями наших товаров.

Создание базы данных

Для хранения информации о товарах, представленных в нашем магазине нам потребуется база данных. Поскольку количество товаров в магазине ограничено, а наша информационная система является учебной представляется логичным создать базу данных на основе текстовых файлов.

В этом случае для описания информационного объекта системы используется строка текстового файла. Строка использует символ-разделитель для разделения атрибутов информационного объекта.

Определим информационный объект нашей системы. По всей видимости это будет «Товар». Ограничим набор свойств (атрибутов) товара следующим списком:

- код товара (число)
- название товара (текст, длина до 250 символов)
- цена товара (число с плавающей запятой, не может иметь отрицательное значение)
- описание товара (текст, длина до 2048 символов)

Кроме того, для лучшего представления товара, введем такой «атрибут» как изображение товара. Фотографии будут храниться в папке data/photos в формате файлов JPEG. Имя файла представляет собой комбинацию <код товара>.jpg.

Данные о товарах будем хранить в текстовом файле catalog.txt. Файл должен находиться в папке data. В качестве символа-разделителя используем символ вертикальной черты: |. Текстовый файл должен иметь кодировку Windows-1251.

Заполнение базы данных

Необходимо вручную произвести начальное наполнение базы данных товаров. Целесообразно ввести данные о 5-7 товарах. Таким образом, файл

catalog.txt будет содержать 5-7 строк. Кроме того, необходимо разместить файлы с изображениями товаров в каталог data/photos.

Вывод списка товаров

Необходимо написать скрипт, который выводит пользователю список товаров вашего магазина. Скрипт должен располагаться в файле catalog/index.phtml. Вывод должен осуществляться в следующем формате:

Код товара	**Наименование товара**	**Цена**
1001	Жесткий диск 40Гб	1000.00

Наименование товара должно быть ссылкой, указывающей на скрипт product.phtml следующим образом:

product.phtml?id=1001

Скрипт product.phtml должен вывести подробную информацию о товаре, включая описание и изображение товара.

Отчет

По окончании работы студенты готовят отчет и предоставляют его преподавателю в электронном виде в формате PDF. Отчет должен разделы: цель работы, порядок выполнения, основная часть должна содержать скриншоты страниц мини-сайта в броузере, текст HTML-кода страниц. Отчет должен заканчиваться выводом.

Элементы языка HTML для создания интерактивных форм

FORM

Используется для создания заполяемой формы. Необходимо присутствие начального и конечного тегов. Внутри элемента FORM разрешается использовать большинство HTML-элементов.

Атрибуты:

NAME – определяет имя формы, уникальное для данного документа. Используется, если в документе присутствует несколько форм.

ACTION – обязательный атрибут. Определяет URL, по которому будет отправлено содержимое формы – путь к скрипту сервера, обслуживающему данную форму.

METHOD – определяет способ отправки содержимого формы. Возможные значения GET (по умолчанию) и POST.

ENCTYPE – определяет способ кодирования содержимого формы при отправке. По умолчанию используется "application/x-www-form-urlencoded".

TARGET – определяет имя окна, в которое возвращается результат обработки отправленной формы. Возможные значения : _self, _parent, _top, _blank или явно указанное имя окна. Подробное описание значений смотрите в атрибуте TARGET элемента A.

Пример:

```
<FORM ACTION=data_post.php METHOD=POST NAME="TestForm">
Фамилия:
<INPUT TYPE="text" name="lastname" SIZE="20" VALUE="Пупкин"><br>
<INPUT TYPE="submit" VALUE="Записать">
</FORM>
```

TEXTAREA

Создает поле для ввода нескольких строк текста. Обычно содержит текст

инициализации, который при загрузке документа изначально будет записываться в данное поле. Элемент TEXTAREA должен располагаться внутри элемента FORM.

Атрибуты:

NAME – обязательный атрибут. Определяет название, которое будет использоваться при идентификации заполненого поля сервером.

ROWS – определяет количество строк текста, видимых на экране.

COLS – определяет ширину текстового поля – в печатных символах.

WRAP – определяет способ переноса слов в заполняемой данной заполняемой форме. Возможные значения:

- off – перенос слов не происходит (значение по умолчанию)
- virtual – перенос слов только отображается, на сервер же поступает неделимая строка.
- physical – перенос слов будет происходить во всех точках переноса.

Пример:

```
<FORM ACTION="data_post.php" METHOD=POST>
<TEXTAREA NAME="address" WRAP="virtual" COLS="40" ROWS="3">Ваш адрес...</TEXTAREA><br>
<INPUT TYPE="submit" VALUE="OK">
</FORM>
```

SELECT

Элемент SELECT создает в заполняемой форме меню типа "Выбор одного пункта из многих" или "Выбор нескольких пунктов из многих". Должен располагаться внутри элемента FORM и иметь как начальный, так и конечный теги. Содержит несколько элементов OPTION, иначе не имеет смысла.

Атрибуты:

MULTIPLE – дает возможность выбора нескольких пунктов меню при удержании клавиши Ctrl. По умолчанию можно выбрать только один пункт

меню.

NAME – определяет имя меню, уникальное для данной формы, которое будет использовать при передаче данных на сервер. Каждый выбранный пункт меню при передаче на сервер будет иметь вид: name/value. Значение (value) формируется элементом OPTION.

SIZE – определяет количество видимых пунктов в меню. Если значение этого атрибута больше единицы, то результатом будет список пунктов.

Пример:

```
<FORM ACTION="receive.cgi">
<SELECT NAME="OS" MULTIPLE>
<OPTION VALUE="DOS">MS-DOS
<OPTION VALUE="WinXP">MS Windows98
<OPTION VALUE="Unix" SELECTED>UNIX
<OPTION VALUE="WinNT">MS Windows NT
</SELECT>
<INPUT TYPE="submit" VALUE="Послать">
</FORM>
```

OPTION

Используется только с элементом SELECT. Элемент OPTION описывает отдельные пункты меню. Не имеет конечного тега.

Атрибуты:

SELECTED – Определяет пункт меню, который будет выбран изначально при загрузке документа. Если меню имеет тип "один из многих", то флагом SELECTED может быть помечен лишь один пунктов меню.

VALUE – Задает данному пункту значение, которое будет использовано наряду с другими сведениями о содержимом заполненной формы. При предоставлении информации на сервер это значение будет объединено со значением атрибута NAME в элементе SELECT.

Пример:

```
<FORM ACTION="script.cgi">
<SELECT NAME="gender">
<OPTION VALUE="male" SELECTED>Мужской
<OPTION VALUE="female">Женский
</SELECT>
<INPUT TYPE="submit" VALUE="OK">
</FORM>
```

INPUT

Элемент INPUT создает поле формы (кнопку, поле ввода, чекбокс и т.п.), содержание которого может быть изменено или активизировано пользователем. Элемент не имеет конечного тега. Элемент INPUT должен располагаться внутри элемента FORM.

Атрибуты:

NAME – определяет имя, используемое при передаче содержания данной формы на сервер. Этот атрибут необходим для большинства типов (атрибут TYPE – см. ниже) элемента INPUT и обычно используется для идентификации поля или для группы полей, связанных логически.

TYPE – определяет тип поля для ввода данных. По умолчанию – это "text". Возможные значения:

- text – создает поле ввода под одну строку текста. Как правило используется совместно с атрибутами SIZE и MAXLENGTH.
- textarea – создает поле ввода для текста в несколько строк. Но для этих целей лучше использовать элемент TEXTAREA
- file – дает возможность пользователю приобщить файл к текущей форме. Возможно использование совместно с атрибутом ACCEPT.
- password – создает поле ввода под одну строку, однако текст, вводимый пользователем, отображается в виде значков "*", скрывая тем самым его

содержание от любопытных глаз.

- checkbox – создает поле ввода для атрибутов типа Boolean ("да"/"нет") или для атрибутов, которые могут одновременно принимать несколько значений. Эти атрибуты представляют собой несколько полей checkbox, которые могут иметь одинаковые имена. Каждое выбранное поле chackbox создает отдельную пару name/value в информации, посылаемой на сервер, даже если результатом будут дублирующиеся имена. Поле этого типа обязательно должно иметь атрибуты NAME и VALUE, а также необязательный атрибут CHECKED, который указывает на то, что поле активизировано.
- radio – создает поле ввода для атрибутов, которые принимают одно значение из нескольких возможных. Все кнопки (radio buttons) в группе должны иметь одинаковые имена, но только выбранная кнопка в группе создает пару name/value, которая будет послана на сервер. Как и для полей checkbox, атрибут CHECKED необязателен; он может быть использован для определения выделенной кнопки в группе кнопок (radio button).
- submit – создает кнопку, при нажатии которой заполненная форма посылается на сервер. Атрибут VALUE в данном случае изменяет надпись на кнопке, содержание которой, заданное по умолчанию, зависит от браузера. Если атрибут NAME указан, то при нажатии данной кнопки к информации, посылаемой на сервер, добавляется пара name/value, указанная для для атрибута SUBMIT, в противном случае пара не добавляется.
- image – создает графическую кнопку-картинку, инициализирующую передачу данных на сервер. Местонахождение графического изображения можно задать с помощью атрибута SRC. При передаче данных серверу сообщаются координаты x и y той точки на изображении, где был произведен щелчок клавишей мыши. Координаты измеряются из верхнего левого угла изображения. При этом информация о поле типа image записывается в виде двух пар значений name/value. Значение name получается посредством добавления к названию соответствующего поля суффиксов ".x" (аб-

сциссы), и ".y" (ординаты).

- reset – создает кнопку, сбрасывающую значения полей формы к их первоначальным значениям. При нажатии кнопки данные на сервер не отправляются. Надпись на кнопке может быть изменена с помощью атрибута VALUE. По умолчанию надпись на кнопке зависит от версии браузера.
- hidden – поля этого типа не отображаются на экране монитора, что позволяет разместить "секретную" информацию в рамках формы. Содержание этого поля посылается на сервер в виде name/value вместе с остальной информацией формы. Этот тип полей удобно использовать для передачи данных от скрипта скрипту незаметно для пользователя.
- button – позволяет создать пользовательскую кнопку в HTML документе, что, при умелом использовании JavaScript, добавляет форме функциональность. Атрибут NAME позволяет задать имя данной кнопке, которое может быть использовано для какой-либо функции в скрипте. Атрибут VALUE позволяет задать текст, который будет отображен на кнопке в документе.

VALUE – задает текстовый заголовок для полей любого типа, в том числе и кнопок. Для таких полей как checkbox или radio, будет возвращено значение, заданное в атрибуте VALUE.

CHECKED – указывает, что поля типов checkbox и/или radio (см. выше атрибут TYPE) активизированы.

SIZE – определяет размер поля в символах. Например, чтобы определить поле с видимой шириной в 24 символа, надо указать SIZE="24".

MAXLENGTH – определяет максимальное количество символов, которые можно ввести в текстовом поле. Оно может быть больше, чем количество символов, указанных в атрибуте SIZE. По умолчанию количество символов не ограничено.

SRC – задает URL-адрес картинки, используемой при создании графической кнопки. Используется совместно с атрибутом TYPE="image".

ALIGN – определяет способ вертикального выравнивания для изображе-

ний. Используется совместно с атрибутом TYPE="image". Полностью аналогичен атрибуту ALIGN элемента IMG. По умолчанию имеет значение bottom.

ACCEPT – конкретизирует тип файла. Используется только совместно с параметром TYPE="file". Значение задается в виде MIME-типа.

Пример 1:

```
<FORM NAME="Form1" ACTION="http://www.igf.ru/cgi-bin/banya.pl">
<INPUT TYPE="hidden" NAME="info" VALUE="Запись в баню на
воскресенье">
<INPUT TYPE="radio" NAME="sex" VALUE="Male" CHECKED> Мужик<BR>
<INPUT TYPE="radio" NAME="sex" VALUE="Female"> Баба<BR>
Имя:<BR>
<INPUT TYPE="text" NAME="textfield" VALUE="Вася Пупкин" SIZE="30"
MAXLENGTH="60"><BR>
Пароль:<BR>
<INPUT TYPE="password" WIDTH="10" NAME="passwd"><BR><BR>
<INPUT TYPE="submit" VALUE="Записать">
</FORM>
```

Пример 2:

```
Хочу получать следующие издания:<br>
<FORM NAME="Form2" ACTION="http://www.igf.ru/cgi-bin/magazines.pl">
<INPUT TYPE="checkbox" NAME="m1">Страшная газета<br>
<INPUT TYPE="checkbox" NAME="m2">6 соток<BR>
<INPUT TYPE="checkbox" NAME="m3" CHECKED>Мурзилка<BR>
<INPUT type="image" src="/img/button.gif" WIDTH="60" HEIGHT="30">
</FORM>
```

Загрузка файлов на сервер

PHP способен принимать файл загружаемый при помощи любого браузера, поддерживающего стандарт RFC-1867 (в том числе Netscape Navigator 3 и выше, Microsoft Internet Explorer 3 с патчем от Microsoft или более поздние версии без патча). Это дает возможность загружать как текстовые, так и бинарные файлы.

Страница для загрузки файлов может быть реализована при помощи специальной формы, которая выглядит примерно так:

Форма для загрузки файлов

```
<form enctype="multipart/form-data" action="_URL_" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="30000" />
Отправить этот файл: <input name="userfile" type="file" />
<input type="submit" value="Send File" />
</form>
```

В приведенном выше примере "_URL_" необходимо заменить ссылкой на PHP-скрипт. Скрытое поле MAX_FILE_SIZE(значение необходимо указывать в байтах) должно предшествовать полю для выбора файла, и его значение является максимально допустимым размером принимаемого файла. Также следует убедиться, что в атрибутах формы вы указали enctype="multipart/form-data", в противном случае загрузка файлов на сервер выполняться не будет.

Переменные, определенные для загруженных файлов, зависят от версии PHP и текущей конфигурации. Суперглобальный массив $_FILES доступен начиная с PHP 4.1.0. Массив $HTTP_POST_FILES доступен начиная с PHP 4.0.0. Эти массивы содержат всю информацию о загруженных файлах. Использование $_FILES является более предпочтительным. В случае, если конфигурационная директива register_globals установлена значением on, дополнительно будут объ-

явлены переменные с соответствующими именами. Начиная с версии 4.2.0 значением по умолчанию для опции register_globals является off.

Содержимое массива $_FILES для нашего примера приведено ниже. Обратите внимание, что здесь предполагается использование имени userfile для поля выбора файла, как и в приведенном выше примере. На самом деле имя поля может быть любым.

$_FILES['userfile']['name'] - Оригинальное имя файла на компьютере клиента.

$_FILES['userfile']['type'] - Mime-тип файла, в случае, если браузер предоставил такую информацию. Пример: "image/gif".

$_FILES['userfile']['size'] - Размер в байтах принятого файла.

$_FILES['userfile']['tmp_name'] - Временное имя, с которым принятый файл был сохранен на сервере.

$_FILES['userfile']['error'] - Код ошибки, которая может возникнуть при загрузке файла. Ключ ['error'] был добавлен в PHP 4.2.0

В случае, если register_globals установлена значением on в конфигурационном файле php.ini, будут доступны дополнительные переменные. Например, $userfile_name будет эквивалентна переменной $_FILES['userfile']['name'], а $userfile_type соответствует $_FILES['userfile']['type'], и так далее. Не стоит забывать, что начиная с PHP 4.2.0 для директивы register_globals значение по умолчанию off. Рекомендуется не полагаться на значение этой директивы.

По умолчанию принятые файлы сохраняются на сервере в стандартной временной папке до тех пор, пока не будет задана другая директория при помощи директивы upload_tmp_dir конфигурационного файла php.ini. Директорию сервера по умолчанию можно сменить, установив переменную TMPDIR для окружения, в котором выполняется PHP. Установка переменной TMPDIR при помощи функции putenv() внутри PHP-скрипта работать не будет. Эта переменная окружения также может использоваться для того, чтобы удостовериться, что

другие операции также работают с принятыми файлами.

Проверка загружаемых на сервер файлов

Для получения более детальной информации вы можете ознакомится с описанием функций is_uploaded_file() и move_uploaded_file(). Следующий пример принимает и обрабатывает загруженный при помощи формы файл.

```
<?php
$uploaddir = '/var/www/uploads/';
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);


print "<pre>";
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    print "File is valid, and was successfully uploaded. ";
    print "Here's some more debugging info:\n";
    print_r($_FILES);
} else {
    print "Possible file upload attack!  Here's some debugging info:\n";
    print "Possible file upload attack!  Дополнительная отладочная информация:\n";
    print_r($_FILES);
}
print "</pre>";
?>
```

В случае, если при отправке формы файл выбран не был, PHP установит переменную $_FILES['userfile']['size'] значением 0, а переменную $_FILES['userfile']['tmp_name'] - пустой строкой. none.

По окончанию работы скрипта, в случае, если принятый файл не был переименован, или перемещен он будет автоматически удален из временной папки.

Замена символов в строке с помощью регулярных выражений

Для замены символов в строке на другие (например в случае присутствия «нежелательных» символов) можно использовать следующие конструкции на языке PHP:

```
$str = ereg_replace(«\n», «<br>», $str);

$str = ereg_replace(«\n\n», «<p>», $str);


$str = ereg_replace(«<br>», «\n», $str);

$str = ereg_replace(«<p>», «\n\n», $str);
```

Лабораторная работа №3. Создание системы администрирования

Цель работы

В рамках выполнения данной лабораторной работы студенты создают систему администрирования нашим мини-магазином. Система должна позволять в интерактивном режиме управлять каталогом товаров — создавать новые записи, изменять существующие и удалять ненужные.

Задание на лабораторную работу

Спланировать приложение системы администрирования. Определить ее функции и структурную схему.

Создать необходимые скрипты и проверить их работу.

Дополнить каталог товаров новыми записями с помощью системы администрирования, доведя число товаров в мини-магазине до 20 — 50 (в зависимости от направления деятельности магазина).

Порядок выполнения работы

Выполнение лабораторной работы производится каждым студентом индивидуально. PHP-скрипты располагаются на web-сервере кафедры электронных приборов в каталоге stdweb. Доступ к серверу осуществляется по протоколу ftp. Логин для входа на сервер и пароль предоставлются преподавателем.

Планирование системы администрирования

Прежде всего необходимо определить набор функций который будет выполняться системой. Краткий список будет выглядеть следующим образом:

- создание новых записей о товарах;

- изменение существующих записей о товарах;
- удаление записей о товарах.

Определим набор скриптов необходимых для выполнения поставленной задачи. Итак, скрипты располагаются в каталоге admin, предлагается следующий состав:

- index.phtml - «главная» страница раздела администрирования, здесь необходимо расположить меню раздела и вывести список товаров снабдив их ссылками «изменить», «удалить», ссылки должны указывать на соответствующие скрипты.
- data_edit.phtml — страница предназначена для создания и изменения записей, содержит форму с полями описывающими товар, форма вызывает для обработки скрипт data_post.php
- data_post.php — скрипт осуществляющий запись новой информации в нашу БД или изменение существующей, послде выполнения своей работы либо выдает сообщение об ошибке либо передает управление «главной» странице раздела
- data_del.php — скрипт осуществляющий удаление ненужных записей, после выполения работы передает управление «главной» странице раздела

Создание набора необходимых скриптов

Определим то, как должна выглядеть форма с полями, описывающая продукт. Очевидно что форма должна передавать данные по методу POST, в поле ACTION необходимо указать URL скрипта обработки: data_post.php. В форме необходим следующий набор полей:

- «код товара» - поле типа INPUT TYPE=TEXT предназначено для внесения информации о коде (артикуле) товара
- «наименование товара» - поле типа INPUT TYPE=TEXT предназначено для внесения информации о наименовании товара
- «цена товара» - поле типа INPUT TYPE=TEXT предназначено для внесения информации о цене товара

- «описание товара» - поле типа TEXTAREA многострочный ввод данных, предназначено для внесения информации описания товара
- «фото товара» - поле типа INPUT TYPE=FILE предназначено для загрузки на сервер изображения (фото) товара
- «кнопка записать» - элемент управления типа INPUT TYPE=SUBMIT , по нажатию на эту кнопку происходит передача управления и данных скрипту data_post.php для последующей обработки.

Алгоритм работы скриптов обработки data_post.php и data_del.php студенты выбирают самостоятельно по согласованию с преподавателем.

Обращаем особое внимание на тот факт, что каждый продукт в нашей базе данных должен занимать одну строку, а элемент TEXTAREA дает результат, который может состоять из нескольких строк разделенных символом перевода строки **\n**.

Заполнение базы данных

После создания необходимых скриптов и их успешной отладки необходимо произвести окончательное наполнение базы данных товаров в интерактивном режиме (с помощью созданных форм и скриптов). Необходимо довести список товаров до 20-50 позиций (в зависимости от характера деятельности магазина).

Отчет

По окончании работы студенты готовят отчет и предоставляют его преподавателю в электронном виде в формате PDF. Отчет должен разделы: цель работы, порядок выполнения, основная часть должна содержать скриншоты страниц мини-сайта в броузере, текст HTML-кода страниц. Отчет должен заканчиваться выводом.