Лабораторная работа № 6. Установка web-сервера

1. Цель работы

Получить теоретические и практические навыки по работе с веб-сервером.

2. Теоретическая часть

Основа Интернета известна всем – это сервера. Именно на них размещены домашние страницы рядовых пользователей, состоящие из нескольких страничек, тематические ресурсы, состоящие из сотен страниц, часто генерируемых динамически и в большинстве своем поддерживаемые группой людей, а также коммерческие проекты и вполне настоящие, несмотря на свою виртуальность, магазины. Большинству рядовых пользователей свой сервер совершенно не нужен. Посудите сами: даже если не выключать домашний компьютер круглосуточно, установив серверное ПО, то для доступа к нему также необходимо круглосуточное соединение с провайдером. Намного проще разместить свою страничку на сервере провайдера. Другое дело – корпоративные пользователи. Как правило, у них уже есть выделенная линия, соединяющая с Интернетом локальную сеть, и выделенный компьютер, выполняющий роль шлюза между локальной и глобальной сетями. На нем-то и можно установить веб-сервер. Или для того, чтобы отдать дань моде, или развернуть крупный проект, приносящий прибыль.

Для того чтобы в Интернете появился сайт, он должен быть размещен на сервере хостера или вашем собственном, подключенном к Сети и имеющем выделенный IP-адрес. Сервер представляет собой компьютер, на котором установлено специальное программное обеспечение, которое тоже называют "веб-сервером".

Веб-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, обычно вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными. Веб-серверы — основа Всемирной паутины.

Клиенты получают доступ к веб-серверу по URL адресу нужной им веб-страницы или другого ресурса.

Дополнительными функциями многих веб-серверов являются:

- ведение журнала обращений пользователей к ресурсам,
- аутентификация пользователей,
- поддержка динамически генерируемых страниц,
- поддержка HTTPS для защищённых соединений с клиентами.

Что должен делать Web-сервер?

Основное действие конечного пользователя в Интернете – это "переход на Web-страницу". На самом общем уровне это предполагает совместную работу пары приложений:

- Web-браузера, такого как Firefox или Internet Explorer, который показывает в удобной для человеческого восприятия форме запрашиваемую страницу, которую он получает от…
- Web-сервера, находящегося, как правило, на удалённой машине, который отвечает

на запрос страницы потоком данных в формате HTML или аналогичном. С браузерами имеют дело Web-пользователи, которые подходят к их выбору и анализу с надлежащей тщательностью. Напротив, серверы видны только техническому персоналу сайтов.

Web-сервер оценивается по целому ряду важнейших параметров:

- **Эффективность**: как быстро он отвечает на запрос?
- **Масштабируемость**: продолжает ли сервер работать надёжно, когда к нему одновременно обращаются много пользователей?
- **Безопасность**: совершает ли сервер только те операции, которые должен? Какие возможности он предлагает для аутентификации пользователей и шифрования потока обмена информацией? Делает ли его использование более уязвимыми соседние приложения или хосты?
- **Работоспособность**: какие у сервера режимы отказа и аварийные ситуации?
- **Соответствие стандартам**: поддерживает ли сервер соответствующие RFC?
- **Гибкость**: можно ли настроить сервер для принятия большого количества запросов или динамических страниц, требующих значительных вычислений, или сложной аутентификации, или ...?
- **Требования к платформе**: на каких платформах возможно использование сервера? Предъявляет ли он особые требования к аппаратной платформе?
- **Управляемость**: легко ли установить и обслуживать сервер? Совместим ли он с организационными стандартами по ведению журналов, аудиту, оценке затрат и т.д.?

Все языки программирования, используемые при разработке web-сайтов, можно разделить на две большие группы.

К *первой* относятся те из них, код которых выполняется на компьютере посетителя сайта, т. е. в браузере, запущенном на компьютере пользователя. Это известные всем JavaScript иVBScript. Программы на этих языках встраиваются в код web-страниц или выносятся в отдельный файл, обращение к которому осуществляется из web-страницы (в этом случае браузер все равно обрабатывает такие «вынесенные» программы таким же образом, как если бы они были встроены в код страницы).

Во *вторую* группу включаются те языки, программы на которых выполняются на том компьютере, где расположен web-сервер. Эта группа более обширна - дело в том, что в принципе на web-сервере могут исполняться программы на любом языке, даже командных `.bat`-файлов MS-DOS, важно лишь, чтобы на нем была установлена программа-интерпретатор этого языка, удовлетворяющая стандарту CGI, которому также должен удовлетворять сам web-сервер.

PHP относится ко второй группе - программа на PHP исполняется на web-сервере. Однако от других CGI-языков PHP сильно отличается в лучшую сторону прежде всего своей простотой. При создании программы на PHP нет необходимости учитывать все те многочисленные мелочи, которые отравляют жизнь программистам на Perl или C++, - не надо заботиться о правах доступа к файлам сценария, не надо прописывать точные пути к различным модулям, нет необходимости следить за отсутствием в файле скрипта недопустимых символов.

Синтаксис языка PHP допускает его легкое освоение как начинающим программистом, так и тем, кто уже использовал ранее какой-либо язык

программирования. Можно целиком и полностью сосредоточиться на решаемой задаче и не думать о мелочах. Именно это и делает PHP подходящим выбором для web-дизайнера, который, начав его использовать, может вообще забыть о каких-либо других CGI-языках. (Впрочем, если исходить из механизма действия, то PHP более правильно называть не "CGI-языком", а препроцессором - что, собственно, отражено даже в его названии.

В то время как CGI-приложение просто выдает некие данные в браузер посетителя, препроцессор просматривает все или некоторые файлы, выдаваемые web-сервером посетителю, и ищет в них определенные команды, которые и выполняет. Именно такой способ работы и позволяет указывать код программ на PHP непосредственно в тексте web-страниц.)Одним из наиболее заметных достоинств PHP является возможность без особых затруднений работать с серверами баз данных.

Возможности PHP можно весьма серьезно расширить с помощью дополнительных модулей, содержащих различные функции. Эти модули при необходимости размещаются на web-сервере, на котором установлен PHP-интерпретатор. Большое количество готовых модулей можно загрузить с адреса http://www.php.net, там же в разделе документации приведено и полное их описание.

Например, модуль Zlib позволяет работать из программы на PHP с архивами в формате Gzip, а модуль libswf - с Flash-презентациями, создавая и редактируя их прямо из программы на PHP.

История создания PHP

В отличие от многих других языков программирования, PHP был создан не какой-либо корпорацией или гением-программистом, а обычным пользователем, Расмусом Лердорфом, в далеком 1994 году. Цель разработки языка была проста - сделать домашнюю страничку Расмуса более интерактивной, а значит, и более привлекательной для посетителей. Расмусразработал базовый синтаксис и написал первый интерпретатор своего языка, получившего название Personal Home Page Tools - т. е. PHP. Этот интерпретатор мог обрабатывать лишь несколько основных команд, однако начало было положено.

В 1995 году Расмус доработал интерпретатор PHP, соединив его с другой своей программой, умевшей обрабатывать HTML-формы (именовавшейся FI - от "Form Interpretator"), а также сделал так, что интерпретатор, получивший название PHP/FI Version 2, мог становиться частью web-сервера. Это новшество позволило программам на PHP исполняться очень быстро. Кроме того, в том же 1995 году интерпретатор PHP был дополнен возможностями1 обработки новых команд, в частности, команд для работы с серверами баз данных и автоматического создания gif-файлов (последнее, к примеру, может быть использовано для генерации кнопок-счетчиков посещений). PHP/FI был

размещен в Сети для всеобщего использования, и началось его повсеместное распространение.

К концу 1997 года PHP использовался более чем на пятидесяти тысячах сайтов. Web-мастера быстро оценили достоинства нового языка web-программирования, такие как легкость освоения и богатство возможностей, и вскоре традиционные Perl и C стали сдавать свои позиции.

Так как исходный код интерпретатора был открыт (а сам интерпретатор, понятно, бесплатен), то энтузиасты стали заниматься его доработкой, и летом 1998 года появился на свет PHP3 - детище
Зива Сураски и Энди Гутманса (Zeev Suraski and Andi Gutmans). PHP3 был создан практически "с нуля", так как его авторы сочли код предыдущих версий недостаточно эффективным. Кроме того, PHP3 стал весьма легко расширяемым продуктом. Любой, создавший на основе определенных стандартов модуль расширения PHP, позволяющий, скажем, работать с архивами какого-либо типа, мог этот модуль интегрировать с программными файлами PHP без каких-либо серьезных затрат времени и сил.

Уже к концу 1999 года число сайтов, построенных на основе PHP, перевалило за миллион. Весьма важным достоинством PHP также являлось то, что программы, позволявшие обрабатывать команды PHP, были созданы практически для всех операционных систем, от Windows до Unix и Linux.

В 2000-м году вышла разработанная компанией Zend Technologies четвертая версия интерпретатора PHP, дополненная множеством новых функций.

В настоящее время именно она является наиболее распространенной - PHP используется более чем на 20% сайтов Сети. Сейчас готова и проходит апробацию уже пятая версия данного языка.

Синтаксис PHP

Синтаксис PHP во многом заимствован из таких языков как C, Java и Perl. Файл, обрабатываемый сервером как правило имеет расширение php.

Поэтому, если вы знакомы хотя бы с одним из них - вам не составит особого труда просто сесть и начать писать программы на PHP.

PHP-код включаются в html-код в следующем виде:

```
<?PHP  текст_кода  ?>
```

или

```
<?
  текст_кода;
?>
```

Коментарии

PHP поддерживает комментарии 'C', 'C++' и оболочки Unix. Например:

```
<?php echo "This is a test"; // Это однострочный комментарий в стиле
c++

/* Это многострочный комментарий,
это ещё одна его строка */

echo "This is yet another test"; echo "One Final Test";
# Это комментарий в shell-стиле ?>
```

echo

<?php echo "Эта информация будет выведена в HTML";?>

Присвоение значений переменным

Переменные в программах на PHP, отделяются символами $.

$city = "Tula";

city - переменная

Tula - значение

Некоторые операции

```
инкремента/декремента;
++$a Pre-increment Увеличивает $a на 1,затем возвращает $a.
$a++ Post-increment Возвращает $a, затем увеличивает $a на 1.
--$a Pre-decrement Уменьшает $a на 1, затем возвращает $a.
$a-- Post-decrement Возвращает $a, затем уменьшает $a на 1.

арифметические:
$a + $b Сложение Сумма $a и $b.
$a - $b Вычитание Разность $a и $b.
$a * $b Умножение Произведение $a и $b.
$a / $b Деление Частное от деления $a на $b.
$a % $b Modulus Целочисленный остаток от деления $a на $b.

строковые:
Имеются две строковые операции. Первая - операция ('.'), которая
возвращает объединение из правого и левого аргументов. Вторая -
операция присвоения ('.='), которая присоединяет правый аргумент в
левому аргументу.

$a = "Hello "; $b = $a . "World!"; // теперь $b содержит "Hello World!"
$a = "Hello "; $a .= "World!"; // теперь $a содержит "Hello World!"
```

Выражения сравнения

Выражения сравнения 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.

сравнения:

```
$a == $b равно TRUE, если $a равно $b.
$a != $b не равно TRUE, если $a не равно $b.
$a <> $b не равно TRUE, если $a не равно $b.
$a < $b меньше TRUE, если $a строго меньше $b.
$a > $b больше TRUE, если $a строго больше $b.
$a <= $b меньше или равно TRUE, если $a меньше или равно $b.
$a >= $b больше или равно TRUE, если $a больше или равно $b.
```

Некоторые операторы

include "имя файла"

- команда для включения содержимого одного файла в другой. Содержимое файла, имя которого указывается в команде, целиком и полностью вставляется на то место, где располагается эта команда, при этом все коды PHP, содержащиеся во вставляемом файле, исполняются так же, как если бы они были на месте этой команды. (Помните, что файл именно вставляется - т. е., например, пути к картинкам, которые должны присутствовать во вставляемом файле, следует указывать от местонахождения того файла, в котором находилась команда include.) Если файл, включаемый в страницу при помощи команды include, отсутствует, то вместо него размещается уведомление об этом, а программа на PHP выполняется дальше. (При необходимости завершения обработки и выдачи web-страницы в случае отсутствия включаемого файла, вместо команды includeследует использовать команду require.)

mail ("Кому", "Тема", "Текст сообщения", "Дополнительные заголовки")

- отправка почтового сообщения. При выполнении данной команды на сервере в соответствии с указанными параметрами формируется электронное письмо и отправляется с помощью установленной на сервере почтовой программы. В качестве параметра "Кому" может выступать набор адресов, разделенных запятыми. "Дополнительные заголовки" могут быть любые (естественно, допустимые почтовыми протоколами!), разделяться они должны должны комбинацией символов /n, которая в PHP означает перевод строки. (Если среди "Дополнительных заголовков" не указано поле From, то оно заполняется по умолчанию почтовой программой web-сервера, например, именем "Unprivileged User".)

echo ("текст")

- вывод на web-страницу какого-либо текста. Чтобы вывести на web-страницу значение какой-либо переменной, достаточно просто написать ее имя внутри выводимой строки: команда echo "это цифра $a" выведет в web-страницу текст "это цифра 1", если ранее переменной $a было присвоено значение, равное единице. В случае необходимости использовать в выводимой строке кавычки или иные специальные символы перед этими символами следует ставить символ " .

if (условие) {...команды, которые должны выполняться, если условие верно...;} else {...команды, которые должны выполняться, если условие неверно...}
-команда, позволяющая выполнить то или иное действие в зависимости от истинности верности или ложности того или иного условия. В фигурных скобках может располагаться несколько команд, разделенных точкой с запятой.
for (начальное значение счетчика, условие продолжения цикла, изменение счетчика на каждом цикле) { ...команды... ;}
- цикл, т. е. повторение указанных в нем команд столько раз, сколько позволит условие изменения счетчика цикла (т. с. переменной, специально выделенной для подсчета числа выполнений команд цикла).
while (условие) { ...команды... }
- цикл с условием. Команды в фигурных скобках выполняются до тех пор, пока выполняется условие в заголовке цикла. Для того чтобы цикл прервался, нужно, чтобы условие выполняться перестало - поэтому внутри цикла необходимо предусмотреть возможность влиять на это условие.
Цикл do {. . .команды. . . } while (условие)
работает так же, однако команды, указанные в фигурных скобках, будут выполнены по меньшей мере один раз - даже если условие выполняться не будет. Прервать выполнение любого цикла можно оператором break -дальнейшее выполнение программы пойдет с команды, следующей после закрывающей фигурной скобки. Оператор жecontinue прерывает текущую стадию выполнения цикла, т. е. после этого оператора дальнейшее выполнение программы начнется с очередной проверки условия заголовка цикла.
switch (выражение) {case значение: ... команды...; break; case другое значение: ... команды...; break;}
- оператор выбора. При его работе содержимое, заключённое в фигурные скобки, просматривается сверху вниз. Как только будет найден оператор case со значением, совпадающим со значением выражения, PHP начнёт выполнять весь код, следующий за этим оператором case до последней фигурной скобки оператора switch или до первого оператора break, в зависимости от того, что появится раньше. (Обратите внимание, что если команду break не указать в конце кода, относящегося к одному варианту значения выражения в заголовке оператора switch, PHP будет выполнять код дальше - т. е. тот, который принадлежит уже следующему оператору case! Это - одно из отличий данного оператора от аналогичных в других языках программирования.) В конце оператора switch можно указать оператор default. Код, стоящий после него, выполнится в том случае, если значение выражения в заголовке оператора не совпадет ни с одним из значений после операторов case.
foreach (переменная as массив) { . . .команды. .. ;}
- поочередное считывание всех элементов массива. Foreach считывает в указанную в его параметрах переменную поочередно все элементы

указанного в них же массива, выполняя каждый раз указанный в фигурных скобках код, в котором может использоваться указанная переменная. (Значения элементов массива этим оператором только считываются, их модификация при помощи
команды foreach невозможна.)

Оператор foreach может быть использован только в PHP версии 4.0 и выше.

Программа на PHP может прерываться кодом web-страницы - для этого достаточно вставить закрывающий тэг до этого кода и открывающий - после. Все, что находится между ними, будет выдаваться в браузер без какой-либо обработки, рассматриваясь как выводимое с помощью команды echo. Иными словами, код

```
<?php if ($a==l) { ?><p>Переменная a равна 1</p><?php> }?>

эквивалентен коду

<?php if ($a==l) {echo "<p> Переменная a равна 1</p>";}?>
```

Однако, первый вариант меньше нагружает процессор компьютера, на котором расположен интерпретатор PHP. Из сказанного также следует, что все программы на PHP, расположенные на одной web-странице, представляют собой одну большую программу, несмотря на то, что они разделяются блоками обычного текста страницы. Именно поэтому переменная, объявленная в расположенном в начале страницы коде, сохраняет свое значение не только до ее конца, но и во всех присоединяемых с помощью команды include файлах.

Пример

```
<?
$rows=5;
$columns=3;
echo '<html><body>';
echo '<table border="1">';
for ($i=1;$i<=$rows;$i++){
echo '<tr>';
 for ($j=1;$j<=$columns;$j++)
{
if ((($i+$j) % 2)==0)
{$color="#000000";}
else
{$color="#ffffff";}

echo "<td bgcolor=$color>$i,$j</td>";
}/* end of for $j*/
echo '</tr>';
}/* end of for $i*/
echo '</table>';
echo '</body></html>';
?>
```

Figure: Пример php.

3. Групповое задание

Требования к выполнению индивидуального задания по лабораторной работе:

- минимум 2 полей формы;
- минимум 1 поля для валидации;
- валидация должна проводиться на стороне сервера;
- результат валидации должен быть возвращен на форму JSON-строкой;
- при возникновении ошибки ввода данных необходимо сообщить пользователю описание ошибки;
- все поля формы должны сохраняться в файл на сервере.

Требуется разработать форму сбора статистических данных о погоде.

4. Варианты для индивидуальных заданий

Требования к выполнению индивидуального задания по лабораторной работе:

- минимум 5 полей формы;
- минимум 1 поле с выпадающим списком;
- минимум 2 поля для валидации;
- валидация должна проводиться на стороне сервера;
- результат валидации должен быть возвращен на форму JSON-строкой;
- при возникновении ошибки ввода данных необходимо сообщить пользователю описание ошибки;
- все поля формы должны сохраняться в файл на сервере.

1) Форма осмотра пациента у врача.
2) Форма заказа мебели на сайте.
3) Форма заказа доставки еды на дом.
4) Форма результата осмотра ноутбука в сервисном центре.
5) Форма для приема драгоценности в ломбард.
6) Форма заказа одежды на сайте.
7) Форма заказа путёвки на сайте туристического агентства.
8) Форма регистрации мобильного телефона на сайте производителя.
9) Форма для отправки объявления в газету.
10) Форма отправки отзыва для гостевой книги.
11) Форма регистрации дорожно-транспортного происшествия.
12) Форма регистрации пользователя социальной сети.
13) Форма заказа оформление букета цветов.
14) Форма регистрации домашнего животного в ветеринарной клинике.
15) Форма заказа книги в онлайн магазине.
16) Форма регистрации на конкурс.
17) Форма добавления монеты в коллекцию.
18) Форма заказа праздника.

19) Форма контроля результатов теста по математике.
20) Форма регистрации устройства в локальной сети.
21) Форма регистрации нового космического объекта.
22) Добавление новой картину в коллекцию картинной галереи.
23) Форма регистрации брака.
24) Добавление нового фильма в фильмотеку.
25) Форма заказа лекарства в аптеке.
26) Форма регистрации кредитной карты в банке.
27) Форма медосмотра на получение прав.
28) Форма регистрации транспортного средства.
29) Форма регистрации на авиарейс.
30) Форма регистрации на железнодорожный рейс.