Простейшая программа на PHP.

Конвертация статического контента в динамический.

Лабораторная работа № 3.

ЦЕЛЬ РАБОТЫ

Ознакомление с основами языка программирования *РНР*, его назначением и возможностями, спектра решаемых задач, ограничениями. Получение навыков работы со средой программирования, обучение работы с FTP‐сервером.

Основная задача языка *PHP* – динамически формировать *HTML*‐код страницы или содержимое документов другого типа. При этом в *PHP* нет команд, которые бы добавляли на страницу таблицы, заголовки, блоки, формы или любые другие теги. Нет, все что может сделать с помощью PHP программист для формирования страниц сайта – это вывести в HTML‐код строку (текст), которая уже потом будет интерпретироваться браузером как HTML‐документ. И если эта строка содержит какие‐либо HTML‐теги, они будут обработаны браузером. Практически, программный код выступает в роли HTML‐верстальщика, который динамически, непосредственно перед загрузкой, верстает страницы сайта.

Поэтому крайне важно знать и уметь самостоятельно работать со статическим HTML‐кодом. Если Вы не знаете, что именно необходимо вывести в качестве правильного HTML‐кода средствами PHP, то даже зная в совершенстве теорию алгоритмов, сам язык и имея многолетнюю практику прикладного программирования – результат Вашей работы, а именно динамически формируемый HTML‐код, будет некачественным и ошибочным.

Самое простое что можно сделать на PHP – это преобразовать сразу весь HTML‐код в динамический. Действительно, если создать переменную и записать в нее сразу весь код – то вывод этой переменной создаст требуемую динамическую страницу. Конечно, такое действие вряд ли имеет смысл, но формирование 100% кода средствами PHP – нормальный и часто применяемый подход, например, при программировании CMS. Но на данном этапе этого не требуется, сначала необходимо научиться работать с PHP, использовать его для небольших задач, формировать HTML‐код лишь частично.

Итак, смысл данной лабораторной работы – в замене статического HTML‐кода на PHP‐программу, которая формирует этот же код в виде текста и выводит его. Поэтому, во время работы над заданием, просто ищите в статической верстке те фрагменты, которые следует заменить. Смело внедряйте вместо них PHP‐программы, которые выводят эти фрагменты динамически. В случае дополнительных заданий – выводите не только формируемый контент, но и дополнительный текст согласно условиям лабораторной работы.

ПРОДОЛЖИТЕЛЬНОСТЬ

4 академических часа (2 занятия)

РЕЗУЛЬТАТ РАБОТЫ

Размещенные на Веб‐сервере и доступные по протоколу http документы (страницы сайта) с частично динамически формирующимся контентом.

ДОПОЛНИТЕЛЬНЫЕ ТРАБОВАНИЯ К РАБОТЕ

Тема статического сайта выбирается студентом самостоятельно, информация – копируется из открытых источников или формируется самостоятельно. Дизайн должен быть лаконичным, цвета

– сочетаемыми, размер шрифта (кегль) – читаемым, но не излишне громоздким.

|  |  |
| --- | --- |
| **Статический сайт** | |
| Страниц | Три, макет страниц одинаковый. |
| Меню | В виде ссылок в шапке страницы. Ссылка на текущую страницу выделена цветом шрифта или фона с помощью стиля CSS. Ссылки реагируют на курсор мыши. |
| Заголовок страницы (title) | ФИО и группа студента, номер лабораторной работы. |
| Шапка (header) | Фиксированной высоты и не реагирует на скроллинг. |
| Подвал (footer) | Фиксированной высоты, приклеен к низу страницы |
| Заголовки | Один H1, не менее двух H2. |
| Текст | Не менее 1Кб на страницу, обязательно наличие списка |
| Фотографии | Не менее двух на не менее одной странице |

После внедрения в статический HTML‐код элементов PHP‐скриптов (частичного конвертирования контента в динамический вид) сайт должен вести себя следующим образом.

|  |  |
| --- | --- |
| **Динамический сайт** | |
| Название страницы (TITLE) | Название страницы сохраняется в переменной, которая затем выводится с помощью PHP. |
| Подвал | В подвале страницы размещается надпись: "*Сформировано*  *15.02.2016 в 12‐57:18*" с актуальной датой и временем. |
| Список | Динамически формируется из массива |
| Меню | Адрес, текст ссылок и класс пункта меню формируется в виде **ДВУХ** включений PHP‐кода. При этом они должны быть одинаковы для всех пунктов на каждой странице, за исключением присваиваемых переменным значений. |
| Фотографии | В зависимости от секунды (четная или нечетная) на одно и тоже место загружаются разные фотографии (разные имена файлов). |

РЕКОМЕНДАЦИИ К СТРУКТУРЕ ПРОГРАММЫ

Вторая лабораторная работа носит ознакомительный характер, поэтому ее содержание достаточно просто. Замена заголовка или других фрагментов HTML‐кода вряд ли вызовет затруднение, формирование строки с текущей датой – тоже. Могут вызывать затруднения два вопроса – это загрузка разных фотографий в зависимости от секунды и вывод пункта меню. Рассмотрим первую задачу более подробно.

Пусть имеется две фотографии одинакового размера: "*fotos/foto1.jpg*" "*fotos/foto2.jpg*". Тогда словесное описание алгоритма можно сформулировать следующим образом.

1. Определяем текущую секунду и сохраняем ее в переменной $s.
2. Вычисляем остаток от деления переменной $s на число 2 и сохраняем его в переменной $os.
3. Если в переменной $os хранится ноль – в переменную $name записываем строку "*fotos/foto1.jpg*".
4. Иначе в переменную $name записываем строку "*fotos/foto2.jpg*".
5. Формируем строку, содержащую тег <img> с вычисленным именем файла и другими необходимыми параметрами и выводим ее в HTML‐код браузера.

Предварительно формулировать реализуемый алгоритм на естественном языке очень важно. Любая программа выполняется именно так, как она написана; делает только то, что ей указали делать, а не то что хочется. Поэтому, если вы не понимаете логику работы программы, не можете выразить ее алгоритм простым человеческим языком – добиться ее правильного функционирования будет очень тяжело. Конечно, со временем и с опытом письменное выражения алгоритма для вас не будет необходимостью. И это будет ознаменовать новый этап развития вас как разработчика программ – язык программирования PHP станет для вас естественным. Вы научитесь выражать на нем свои мысли и читать его также просто, как сейчас читаете эти слова. Но пока этого не произошло, формулируйте алгоритм словами, а лишь затем переводите его в программный код. Фактически, это действительно перевод с одного языка, в данном случае русского, на PHP. Сделаем это прямо сейчас!

Листинг 1

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

$s = date('s'); *// определяем текущую секунду*

$os = $s % 2; *// вычисляем остаток от деления*

if( $os === 0 ) *// если в переменной $os хранится ноль*

$name='fotos/foto1.jpg'; *// сохраняем имя первого файла*

else *// иначе*

$name='fotos/foto2.jpg'; *// сохраняем имя второго файла*

echo '<img src="'.$name.'" alt="Меняющаяся фотография">'; *// выводим сформированный HTML‐код*

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

Ура, мы сделали это! Заметьте – при правильном переводе в программе пункты нашего словесного описания преобразуются в комментарии. Но, честно говоря, данный код хоть и решает задачу, но далек от идеала. Оптимизируем его, убрав переменные, которые используются только один раз, а также по‐другому формируя HTML‐код.

Листинг 2

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

if( date('s') % 2 === 0 ) *// если секунда четная*

$name='1'; *// имя файла содержит "1"*

*else // иначе*

$name='2'; *// имя файла содержит "2"*

*// формируем и выводим HTML-код*

echo '<img src="fotos/foto'.$name.'.jpg" alt="Меняющаяся фотография">';

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

Результат работы программы будет абсолютно одинаков с первым примером, но смотрится он более красиво. Не так ли? Для примера сформируем и третий вариант кода, который также может иллюстрировать возможности PHP и наличие у одной задачи разных алгоритмов решения.

Листинг 3

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

echo '<img src="fotos/foto'; *// выводим первую часть формируемого HTML-кода*

echo date('s') % 2 + 1; *// вычисляем переменную часть имени и выводим ее*

echo '.jpg" alt="Меняющаяся фотография">'; *// выводим вторую часть HTML-кода*

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

Или, если сократить запись еще больше, можно просто написать такой код.

Листинг 4

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

*// формируем HTML-код и сразу его выводим*

echo '<img src="fotos/foto'.(date('s') % 2+1).'.jpg" alt="Меняющаяся фотография">';

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

Попробуйте все варианты кода, убедитесь, что результат их работы одинаков. Разберитесь и убедитесь, что вы понимаете каждый СИМВОЛ программы. Если вы не сделаете этого – разбирать более сложные примеры будет просто невозможно. Ведь Вы не умеете "говорить" на PHP, не

знаете его слов, не сможете перевести естественно‐языковое описание алгоритм на язык программирования.

Аналогично поступим и со второй задачей. Пусть в HTML‐коде пункт меню формируется следующим тегом: "<a href="page2.php" class="selected\_menu">Вторая страница</a>". Ясно, что различные пункты меню отличаются адресом, в данном случае "page2.php", и текстом ссылки. Стиль "selected\_menu" будет определять выделенный пункт меню (ссылка без класса – обычный пункт меню). Тогда, для вывода пункта меню можно предложить такую программу.

Листинг 5

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

<?php *// начинаем PHP скрипт*

// формируем и выводим строку с ссылкой

echo '<a href="page2.php" class="selected\_menu">Вторая страница</a>';

?>

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

Сейчас мы просто выводим строку с кодом ссылки с помощью PHP. Но в задании лабораторной работы необходимо делать это с помощью двух фрагментов кода. Нет ничего проще!

Листинг 6

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

<?php *// начинаем первый PHP скрипт*

echo '<a href="page2.php"';

?><?php *// начинаем второй PHP скрипт*

echo ' class="selected\_menu">Вторая страница</a>';

?>

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

Обратите внимание: между окончанием первого фрагмента PHP‐программы и началом второго в данном примере нет ни пробелов, ни переводов строки. Если они будут, то и в HTML‐коде ссылки появится пробел или перенос строки, что будет совершенно лишним. Продолжим приближать наш код к заданию лабораторной работы: сделаем так, что PHP выводит ТОЛЬКО адрес и текст ссылки, а также ее класс.

Листинг 7

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

<a href="<?php *// начинаем первый PHP скрипт*

echo 'page2.php';

?>" class="<?php *// начинаем второй PHP скрипт*

echo 'selected\_menu">Вторая страница';

?></a>

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

Формирование HTML‐кода ссылки будет происходить следующим образом (для краткости уберем комментарии и необязательные переносы строк).

|  |  |
| --- | --- |
| **Шаг** | **1** |
| **Анализируемый фрагмент кода PHP‐страницы** | <a href=" |
| **Результат** | <a href=" |
| **Сформированный HTML‐ код** | <a href=" |

Первая часть исходного кода страницы – статическая. Соответственно PHP‐его не обрабатывает, и она передается в браузер без изменений.

|  |  |
| --- | --- |
| **Шаг** | **2** |
| **Анализируемый фрагмент кода PHP‐страницы** | *<?php echo '*page2.php'; ?> |
| **Результат** | page2.php |

|  |  |
| --- | --- |
| **Сформированный HTML‐ код** | <a href="page2.php |

На втором шаге встречается PHP‐скрипт, при выполнении которого выводится адрес ссылки.

|  |  |
| --- | --- |
| **Шаг** | **3** |
| **Анализируемый фрагмент кода PHP‐страницы** | " class=" |
| **Результат** | " class=" |
| **Сформированный HTML‐ код** | <a href="page2.php" class=" |

Третий фрагмент – опять статический HTML‐код. Обратите внимание: он ежит между двумя скриптами, и если в прошлый раз мы специально оговаривали что меду ними ничего не должно быть, чтобы структура тега ссылки не нарушалась, то сейчас мы намеренно добавили этот фрагмент.

|  |  |
| --- | --- |
| **Шаг** | **4** |
| **Анализируемый фрагмент кода PHP‐страницы** | <?php echo 'selected\_menu">Вторая страница'*; ?>* |
| **Результат** | selected\_menu">Вторая страница |
| **Сформированный HTML‐ код** | <a href="page2.php" class="selected\_menu">Вторая страница |

Четвертый фрагмент – снова PHP скрипт. Он формирует продолжение тега ссылки с классом и текстом ссылки.

|  |  |
| --- | --- |
| **Шаг** | **5** |
| **Анализируемый фрагмент кода PHP‐страницы** | </a> |
| **Результат** | </a> |
| **Сформированный HTML‐ код** | <a href="page2.php" class="selected\_menu">Вторая страница</a> |

Заключительная часть фрагмента – снова статическая, она завершает формирование HTML‐кода ссылки.

Такой последовательный разбор действия, с последовательной записью формируемого HTML‐ кода – хороший способ отладки работающего PHP‐скрипта, когда он формирует "странный" HTML‐ код. Но вернемся к тексту программы: он должен быть универсален для всех пунктов меню на всех страницах. Ясно, что сейчас это далеко не так.

Листинг 8

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

<a href="<?php *// начинаем первый PHP скрипт*

$name='Вторая страница'; *// переменная с текстом ссылки*

$link='page2.php'; *// переменная с адресом ссылки*

$current\_page=true; *// переменная, определяющая активность пункта меню*

echo $link; *// выводим адрес ссылки*

?>"><?php *// начинаем второй PHP скрипт*

if( $current\_page ) *// если пункт меню активный*

echo ' class="selected\_menu"'; *// выводим соответствующий класс*

echo $name*; // выводим текст ссылки*

?></a>

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

Меняя значения переменных в начале первого фрагмента кода можно формировать ссылку с необходимым адресом, текстом и оформлением.

СПРАВОЧНАЯ ИНФОРМАЦИЯ

|  |  |
| --- | --- |
| Оформление начала и окончания  PHP‐кода | <?php .... ?> |
| Комментарии | .... *// комментарий до конца строки*  */\* многострочный комментарий*  *....*  *\*/* |
| Присвоение переменной строкового значения | $A='строка'; |
| Функция вывода даты и некоторые ее параметры | date($f); |
| Вывод значения переменной | echo $A; |
| Вывод строки | echo 'Строка'; |
| Условный оператор | if( <условие> ) *// если условие выполняется*  ... *// выполняется этот оператор*  else *// иначе*  ... *// выполняется этот оператор* |

КОНТРОЛЬНЫЕ ВОПРОСЫ К ЛАБОРАТОРНОЙ РАБОТЕ

Для успешной защиты работы помимо соответствующего требованиям результата необходимо уверенно отвечать на нижеперечисленные и другие вопросы.

1. Что такое PHP?
2. Какой основное назначение PHP?
3. В чем отличие статического и динамического контента?
4. Как внедрить PHP‐код на статическую страницу?
5. Какие общепринятые требования к страницам с PHP кодам?