Московский технологический институт

Проектирование и разработка интернет-приложений № 3-4

(Введение в PHP. Основы клиент-серверного взаимодействия)

Кафедра: Информатики и автоматизации
Автор: к.т.н., доцент Долин Георгий Аркадьевич
Контакты: dolin1974@gmail.com

Москва - 2016

Рекомендуемая литература

- «PHP5. Полное руководство»
  Джон Коггзолл. Диалектика. 2006 г.
- «Разработка Web-приложений с помощью PHP и MySQL»
  Люк Веллинг, Лаура Томсон. Вильямс. 2007 г.
- «AJAX и PHP. Разработка динамических веб-приложений»
  Кристиан Дари, Богдан Бринзаре, Филип Черчез-Тоза, Михай Бусика.
  Символ-Плюс. 2006 г.
- «PHP, MySQL, XML. Программирование для Интернета»
  Елена Бенкен. БХВ-Петербург. 2007 г.
- «Профессиональное программирование на PHP»
  Джордж Шлосснейгл. Вильямс. 2006 г.

Введение в PHP
Базовый синтаксис

PHP — это рекурсивный акроним аббревиатуры PHP Hypertext Preprocessor. Команды на языке PHP обрамляются специальными дескрипторами — тэгами языка PHP. Все, что находится вне этих тегов, игнорируется интерпретатором. Поддерживаются следующие стили написания тэгов:

- XML-стиль (рекомендуемый);
  ```
  <?php код на PHP ?>
  ```
- HTML-стиль;
  ```
  <script language="php"> код на PHP </script>
  ```
- Краткий стиль;
  ```
  <? код на PHP ?>
  ```
- ASP-стиль.
  ```
  <% код на PHP %>
  ```

Введение в PHP
Базовый синтаксис

Существует ряд требований, которые необходимо соблюдать при программировании на PHP:

- Каждая команда заканчивается точкой с запятой (;);
- Одну команду можно записывать в несколько строк или несколько команд в одну строку;
- PHP чувствителен к регистру символов в именах переменных и функций;

```
<?php
    $index = 10;
    print($Index);      // Ошибка
?>
```

Введение в PHP
Базовый синтаксис

- PHP нечувствителен в отношении ключевых слов, к пробелам, переводам строки, знакам табуляции.

Этот код полностью корректен:

```
<?php
    $index = 10;
    $index = 10 + 20;
    $index = 10+10;
    $index =
    10
    +
    10;
?>
```

Введение в PHP
Комментарии

PHP поддерживает три вида комментариев: один многострочный и два однострочных. PHP-парсер никак не анализирует комментарии, они просто игнорируются.

```
<?php
    /*
        Первый
        вид
        комментария
     */

    //  Второй

    #   Третий
?>
```

Введение в PHP
Переменные

- Все имена переменных должны начинаться со знака доллара ($);
- Объявления не являются обязательными. Переменная начинает существовать с момента присвоения ей значения или с момента первого использования. Если использование начинается раньше присвоения, то переменная будет содержать значение по умолчанию;
- Переменной не назначается определенный тип. Тип определяется хранящимся значением и текущей операцией.

Введение в PHP
Переменные

Первым символом после $ должна быть буква или символ подчеркивания. Далее в имени переменной могут присутствовать буквы, цифры и символ подчеркивания.

```
<?php
    $I;                 // Допустимо
    $1;                 // Недопустимо
    $_1 ;               // Допустимо
    $firstName;     // Допустимо
    $7Lucky;            // Недопустимо
    $~password;     // Недопустимо
    $Last!Visit;        // Недопустимо
    $Compute-Mean  ;    // Недопустимо
?>
```

Введение в PHP

Переменные. Пример

```php
<?php
    $foo = 'Bob';                 // Присваивает $foo значение
'Bob'
    $foo = "My name is Mike"; // Изменение $foo
    $bar = 25;                  // Присваивает $bar значение 25
    $bar = 2 + 2;                  // Присваивает $bar 4
    $tmp = $foo;                  // Присваивает $tmp значение $foo
    $tmp = &$foo;                  // Ссылка на $foo через $tmp
    $foo = "John";              // Изменение $foo
    echo $tmp;                  // Выведет на экран "John"
    $foo = "Mike";              // Изменяем значение $foo
    unset($foo);                  // Удаляем переменную $foo
    echo $tmp;                  // Выведет на экран "Mike"
?>
```

Введение в PHP
Предопределенные переменные

$GLOBALS — Массив, содержащий все глобальные переменные.
$_ENV — Массив переменных окружения.
$_COOKIE — Массив файлов cookie, отправленных на сервер.
$_GET — Массив переменных, отправленных методом GET.
$_POST — Массив переменных, отправленных методом POST.
$_FILES — Массив, содержащий информацию о загруженных файлах.
$_REQUEST — Массив, содержащий $_GET,$_POST,$_FILES,$_COOKIE.
$_SESSION — Массив переменных, размещенных в сессиях PHP.
$_SERVER — Массив, содержащий информацию о сервере.

Введение в PHP
Типы данных

PHP поддерживает восемь типов данных.

Четыре скалярных типа:

- **`boolean`** — логический;
- **`integer`** — целое число;
- **`float`** (**`double`**) — число с плавающей точкой;
- **`string`** — строка.

Два смешанных типа:

- **`array`** — массив;
- **`object`** — экземпляр класса.

Два специальных типа:

- **`resource`** — ссылка на внешний по отношению к скрипту источник данных (файл на диске, изображение в памяти и т.п.);
- **`NULL`** — отсутствие какого либо значения.

Введение в PHP

Типы данных. Пример

```php
<?php
    $foo = TRUE;            // Логический
    $int = 1234;            // Целое число
    $flt = 1.234;           // Число с плавающей точкой
    // Это простая строка
    echo "Это простая строка";
    // Это вставит: новую строку
    echo "Это вставит: \n новую строку";
    // Переменная OK вставилась в текст
    $a = "OK"; echo "Переменная $a вставилась в текст";
    // Это простая строка
    echo 'Это простая строка';
    // Он сказал "I'll be back"
    echo 'Он сказал: "I\'ll be back"';
    // Это не вставит: \n новую строку
    echo 'Это не вставит: \n новую строку';
    //Переменная $a не подставляется
    $a = "OK"; echo 'Переменная $a не подставляется';
?>
```

Введение в PHP

Типы данных. Полезные функции

isset (имя_переменной) - сообщает, существует ли переменная.
unset (имя_переменной) - уничтожает указанную переменную
empty (имя_переменной) - сообщает, присвоено ли переменной какое-либо значение.
gettype(имя_переменной) - возвращает тип указаной переменной
settype(имя_переменной, тип) - конвертирует переменную в другой тип.
is_bool(имя_переменной) - проверяет является ли тип переменной логическим.
Функции is_numeric(), is_float(), is_int(), is_string(), is_object(), is_array() работают по аналогии.

Введение в PHP
Константы

Для задания значений, которые не будут меняться в ходе выполнения сценария можно использовать константы. Так же как и переменные, константы могут быть определены и доступны в любом месте сценария, но у них есть и ряд особенностей:

- У констант нет префикса в виде знака доллара;
- Константам нельзя присваивать значения, их можно определить вызовом функции define();
- Константы не могут быть определены или аннулированы после первоначального объявления.

Введение в PHP
Константы. Пример

```php
<?php

    define ('PI', 3.14);
    $index = 10 * PI;      // Верно
    PI = 10 * 3.14;            // Ошибка!

    define("CONSTANT", "Здравствуй, мир.");
    echo CONSTANT;         // Выведет "Здравствуй, мир."
    echo Constant;         // Выведет "Constant" и предупреждение

?>
```

Введение в PHP
Предопределенные константы

__LINE__	- Номер текущей строки.
__FILE__	- Полный путь и имя текущего файла.
__FUNCTION__	- Имя текущей функции.
__CLASS__	- Имя текущего класса.
PHP_EXTENSION_DIR	- Каталог расширений PHP
PHP_OS	- Операционная система
PHP_VERSION	- Версия PHP
PHP_CONFIG_FILE_PATH	- Каталог размещения php.ini

Введение в PHP
Операторы

Операторы бывают трех видов:

1. унарные, те, которые работают только с одним аргументом.
2. бинарные — с двумя.
3. тернарный оператор ?:.

Введение в PHP
Арифметические операции

```
-$a             // Смена знака
$a + $b         // Сумма
$a - $b         // Разность
$a * $b         // Произведение
$a / $b         // Частное
$a % $b         // Остаток от деления
$a += $b        // Аналогично $a = $a + $b
$a -= $b        // Аналогично $a = $a - $b
$a *= $b        // Аналогично $a = $a * $b
$a /= $b        // Аналогично $a = $a / $b
$a %= $b        // Аналогично $a = $a % $b
```

Введение в PHP
Операции сравнения

```
$a == $b        // TRUE если $a равно $b.
$a === $b       // TRUE если $a равно $b И имеет тот же тип
$a != $b        // TRUE если $a не равно $b.
$a !== $b       // TRUE если $a не равно $b ИЛИ у них разные типы.
$a < $b     // TRUE если $a строго меньше $b.
$a > $b     // TRUE если $a строго больше $b.
$a <= $b        // TRUE если $a меньше или равно $b.
$a >= $b        // TRUE если $a больше или равно $b.
```

Введение в PHP
Логические операции

```
$a and $b  // TRUE если и $a, и $b TRUE.

$a or $b   // TRUE если или $a, или $b TRUE.

$a xor $b  // TRUE если $a, или $b TRUE, но не оба.

!$a    // TRUE если $a не TRUE.

$a && $b   // TRUE если и $a, и $b TRUE.

$a || $b   // TRUE если или $a, или $b TRUE.
```

Введение в PHP
Побитовые операции

```
$a & $b // Побитовое И

$a | $b // Побитовое ИЛИ

$a ^ $b // Исключающее или

~ $a       // Отрицание

$a << $b   // Побитовый сдвиг влево

$a >> $b   // Побитовый сдвиг вправо
```

Введение в PHP
Специфичные операции

Конкатенация

```php
$a = "Hello ";

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

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

Подавление ошибки

```php
@$a = 1 / 0;           // Ошибка не будет сгенерирована
```

Введение в PHP
Инкремент / Декремент

```
++$a    // Увеличивает $a на единицу и возвращает значение $a
$a++    // Возвращает значение $a, а затем увеличивает $a на
единицу
--$a    // Уменьшает $a на единицу и возвращает значение $a
$a--    // Возвращает значение $a, а затем уменьшает $a на единицу
```

```
<?php

    $a = 5;
    echo "Должно быть 5: " . $a++ . "<br>";
    echo "Должно быть 6: " . $a . "<br>";
    $a = 5;
    echo "Должно быть 6: " . ++$a . "<br>";
    echo "Должно быть 6: " . $a . "<br>";

?>
```

Введение в PHP
Тернарная операция

По сути является аналогом условной конструкции **if...else**
Она записывается следующим образом:

```
условие ? значение, если условие истинно : значение, если ложно
```

```php
<?php
    $grade = 3;
    $result = ($grade > 2 ? 'Сдал' : 'Не сдал');
    echo $result;
?>
```

Введение в PHP
Демонстрация

- Использование echo и print;
- Использование переменных и констант;
- Использование специальных функций;
- Использование арифметических операторов;
- Использование операторов сравнения;
- Использование логических операторов.

Введение в PHP
Практическая работа

1. Создайте файл 1-1.php, содержащий 5 разных переменных, присвойте переменным значения разного типа. Используя gettype() выведите тип каждой переменной.
2. Создайте файл 1-2.php, содержащий 2 переменные числового типа. Произведите над переменными произвольное арифметическое действие и выведите его результат.
3. Создайте файл 1-3.php, содержащий 2 переменные строкового типа. Инициализируйте переменные произвольным текстом. С помощью конкатенации объедините содержимое переменных и выведите результат.
4. Создайте файл 1-4.php, содержащий 2 переменные с одинаковым типом значений. Используя тернарный оператор сравнения проведите исследование на возвращаемые результаты.

Введение в PHP

Управляющие конструкции

Конструкция if

Указынные дейтвия выполняются тогда и только тогда, когда условие истинно.

```
if(условие){
    Действие;
}
```

```
if($index > 0) {
    echo 'Index > 0';
}
```

Конструкция if...else

Если условие истинно, выполнятся действия из блока if, в противном случае — из блока else.

```
if(условие){
    Действие;
}else{
    Действие;
}
```

```
if($index > 0){
    echo 'Да';
}else{
    echo 'Нет';
}
```

Введение в PHP
Управляющие конструкции

Конструкция elseif

Если условие блока if истинно, выполнятся действия блока if. В противном случае, если условие блока elseif истинно, выполнятся действия блока elseif. Во всех остальных случаях выполнятсе действия из блока else.

```
if(условие){
    Действие;
}elseif(условие){
    Действие;
}else{
    Действие;
}
```

```
if($numb < 5){
    $discount = 0;
elseif($numb >= 5 && $numb <= 10)
    $discount = 5;
else{
    $discount = 10;
}
```

Введение в PHP

Управляющие конструкции

Конструкция switch

Если значение переменной соответствует значению одного из блоков case, выполнятся действия из этого блока. В противном случае - из блока default.

```
switch(Переменная){
    case Значение 1:
        Действие 1;
        [break;]
    case Значение 2:
        Действие 2;
        [break;]
    [default: Действие;]
}
```

```
switch($day){
    case 1:
        echo 'Понедельник';
break;
    case 2:
        echo 'Вторник'; break;
    case 3:
        echo 'Среда'; break;
    case 4:
        echo 'Четверг'; break;
    case 5:
        echo 'Тяпницо'; break;
    case 6:
        echo 'Суббота'; break;
    case 7:
        echo 'Воскресенье';
break;
    default:
        echo 'Нет такого дня';
}
```

Введение в PHP
Циклы

Циклы предназначены для многократного исполнения набора инструкций.

Цикл for

В цикле for указывается начальное и конечное значения счетчика, а так же шаг, с которым счетчик будет изменяться. Изменяться счетчик может как в положительную, так и отрицательную сторону. Действия выполнятся столько раз, сколько итераций пройдет от начального значения счетчика до достижения конечного, с указынным шагом.

```
for(начало;конец;шаг){
    Действие;
    ...
}
```

```
for ($i = 1; $i <= 5; $i++) {
        $sum += $i;
        echo $sum;
    }
```

Введение в PHP
Циклы

Цикл while

Действия будут выполняться до тех пор, пока условие истинно.
Цикл while является циклом с предусловием.

```
while (условие) {
    Действие;
продолжается';
    ...

}
```

```
while ($state == 'Солнце высоко') {
        echo 'Рабочий день

        $state = 'Солнце заходит';

}
```

Цикл do...while

Цикл do...while является циклом с постусловием. Это значит, что сначала будет выполняться действие, а потом проверяться условие. Таким образом действие всегда выполнится минимум один раз.

```
do{
    Действие;
    ...
} while (условие);
```

```
do{
echo 'Пиф-паф';
} while ($state == 'Живой');
```

Введение в PHP
Управление циклами

Break прерывает работу цикла. Интерпретатор перейдет к выполнению инструкций, следующих за циклом.

```
$index = 1;
while ($index < 10){
    echo "$index <br>";
    $index++;
    if ($index == 5)
        break;
}
```

Continue прерывает выполнение текущей итерации цикла. Цикл продолжит выполняться со следующей итерации.

```
$index = 0;

while ($index < 10){
    $index++;
    if ($index == 5)
        continue;
        echo "$index <br>";
}
```

Введение в PHP
Массивы

Массив — это структура, в которой хранится упорядоченный набор данных. Эти данные называются элементами массива. Каждый элемент массива имеет свой уникальный индекс.
В PHP массив можно создать следующими способами:

```php
<?php
    $zoo[0] = 'слон';
    $zoo[6] = 'крокодил';
    $zoo[4] = 'жираф';
    $zoo[] = 'осел';        // Индекс равен 7

    // или

    $zoo = array ('лев', 'медвед', 'обезьяна');
    echo count ($zoo);              // Количество элементов массива
?>
```

Введение в PHP
Ассоциативные массивы

В ассоциативных массивах используется не числовой, а строковый индекс.

```
<?php
    $pets['dog'] = 'Бульдог';
    $pets['cat'] = 'Шиншилла';
    $pets['fish'] = 'Золотая';

    // или

    $pets = array ('lizard' => 'Игуана',
                   'spider' => 'Черная вдова',
                   'parrot' => 'Ара');
    print_r ($pets);       // Печать массива
?>
```

Введение в PHP
Многомерные массивы

Массив называется многомерным тогда, когда в качестве его элементов выступают не только скалярные величины, но и сами массивы.

```php
<?php
    $users = array (
                0 => array (
                                'login' => 'admin',
                                'password' => 'hskdfuegefdjfdg'
                            ),
                1 => array (
                                'login' => 'telo',
                                'password' => 'ppqmcnvkfghye'
                            )
    );
    echo $users[0]['login'];  // admin
?>
```

Введение в PHP
Цикл foreach

Очень удобен при работе с массивами. Указанные действия выполняются для **каждого** элемента массива $array, при этом $key — номер элемента массива $array, $value — значение этого элемента.

```
foreach ($array as [ $key => ] $value){
    Действия;
    ...
}
<?php
    $pets[] = 'Собака';
    $pets[] = 'Кошак';
    $pets[] = 'Рыбка';
    foreach ($pets as $index => $value) {
        echo "Элемент №$index имеет значение: \"$value\"<br>";
    }
?>
```

Введение в PHP
Демонстрация

- Использование if, else, elseif;
- Использование switch;
- Использование for, while, do...while;
- Использование массива;
- Использование ассоциативного массива;
- Использование цикла foreach.

Введение в PHP
Практическая работа

1. Используя условный переход, выведите сообщение «Счастливчик!» если $age попадает в диапазон между 18 и 35. Если значение иное, выведите «Не повезло». Расширьте предыдущую конструкцию сообщением «Слишком молод», если $age в диапазоне между 1 и 17.

2. Используя циклы, сформируйте массив четных чисел из диапазона от 1 до 100. Выводя массив на экран, исключите из вывода все числа, которые не делятся на 5.

3. Создайте массив со следующими элементами: Name, Address, Phone, Mail и заполните его. С помощью цикла foreach осуществите форматированный вывод массива в виде: «элемент: значение».

Основы клиент-серверного взаимодействия

Протокол HTTP

HTTP (HyperText Transfer Protocol, протокол передачи гипертекста) — протокол прикладного уровня для передачи данных в первую очередь в виде текстовых сообщений. Основой протокола HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.
Полное описание протокола содержится в спецификации, опубликованной на сайте http://www.w3.org/protocols или в RFC 2616.

Протокол HTTP
Запрос клиента

Клиент инициирует взаимодействие с сервером и посылает запрос, содержащий:

- метод доступа;
- адрес URI (Uniform Resource Identifier, универсальный идентификатор ресурса);
- версию протокола;
- сообщение с информацией о типе передаваемых данных, информацией о клиенте, пославшем запрос, и, возможно, с содержательной частью (телом) сообщения.

Протокол HTTP
Ответ сервера

Ответ сервера содержит:

- строку состояния, в которую входит версия протокола и код возврата (успех или ошибка);
- сообщение, в которое входит информация сервера, метаинформация (т.е. информация о содержании сообщения) и его тело.

Протокол HTTP
Пример

```
GET / HTTP/1.1 ▯
Host: www.rosnou.ru ▯
Accept: */* ▯
Referrer: http://www.google.ru/search?
aq=f&complete=1&hl=ru&newwindow=1&q=%D0%A0%D0%BE%D1%81%D0%9D%D0%9E
%D0%A3&btnG=%D0%9F%D0%BE%D0%B8%D1%81%D0%BA&lr= ▯
User-Agent: Mozilla 4.0 (compatible; MSIE 6.1,…) ▯
▯
HTTP/1.1 200 OK ▯
Date: Sun, 06 Jan 2008 11:19:28 GMT ▯
Server: Apache/2.2.4 ▯
Pragma: no-cache ▯
Content-Length: 23341 ▯
Content-Type: text/html; charset=windows-1251 ▯
▯
<html> . . .
</html>
```

Протокол HTTP
Передача форм

```
<html>
  <head>
    <title>Отправка формы</title>
  </head>
  <body>
    <form action="handler.php" method="...">
      Логин:  <input type="text" name="login"><br>
      Пароль: <input type="password" name="pass"><br>
      <input type="submit" value="Отправить">
    </form>
  </body>
</html>
```

Протокол HTTP
Обработка форм

После отправки данные попадают в глобальные массивы, имена которых соответствуют названию метода отправки.

- $_POST — если данные переданы методом POST;
- $_GET — если данные переданы методом GET;
- $_REQUEST — если данные были переданы любым из них.

```
<?php
    if(isset($_POST['login']) && $_POST['login'] != '' &&
            isset($_POST['pass']) && $_POST['pass'] != ''){
        echo 'Привет ' . $_POST['login'] . '!';
        echo 'Твой пароль: ' . $_POST['pass'] . '<br>';
    }else{
        echo 'Некорректное имя и пароль!<br>';
    }
?>
```

Протокол HTTP
Методы

OPTIONS - Возвращает методы HTTP, которые поддерживаются сервером.
GET - Запрашивает содержимое указанного ресурса.
HEAD - Аналогичен методу GET, но в ответе отсутствует тело.
POST - Передаёт пользовательские данные заданному ресурсу.
PUT - Загружает указанный ресурс на сервер.
DELETE - Удаляет указанный ресурс.
TRACE - Возвращает полученный запрос так, что клиент может увидеть, что промежуточные сервера добавляют или изменяют в запросе.
CONNECT - Для использования вместе с прокси-серверами, которые могут динамически переключаться в туннельный режим SSL.

Протокол HTTP
Метод GET

```
GET /somepage.php?login=vasya&password=superpass HTTP/1.1 
Host: www.rosnou.ru 
Accept: */* 
Referrer: http://www.google.ru/search?
aq=f&complete=1&hl=ru&newwindow=1&q=%D0%A0%D0%BE%D1%81%D0%9D%D0%9E%
D0%A3&btnG=%D0%9F%D0%BE%D0%B8%D1%81%D0%BA&lr= 
User-Agent: Mozilla 4.0 (compatible; MSIE 6.1,…) 


HTTP/1.1 200 OK 
Date: Sun, 06 Jan 2008 11:19:28 GMT 
Server: Apache/2.2.4 
Pragma: no-cache 
Content-Length: 23341 
Content-Type: text/html; charset=windows-1251 

```

Протокол HTTP
Метод POST

```
POST /somepage.php HTTP/1.1 
Host: www.rosnou.ru 
Accept: */* 
Referrer: http://www.google.ru/search?
aq=f&complete=1&hl=ru&newwindow=1&q=%D0%A0%D0%BE%D1%81%D0%9D%D0%9E%
D0%A3&btnG=%D0%9F%D0%BE%D0%B8%D1%81%D0%BA&lr= 
User-Agent: Mozilla 4.0 (compatible; MSIE 6.1,…) 

login=vasya&password=superpass 


HTTP/1.1 200 OK 
Date: Sun, 06 Jan 2008 11:19:28 GMT 
Server: Apache/2.2.4 
Pragma: no-cache 
Content-Length: 23341 
Content-Type: text/html; charset=windows-1251 

```

Протокол HTTP
Основные заголовки

Accept. Данный заголовок предназначен для информирования сервера о типах данных, поддерживаемых браузером. Перечисление идет через запятую. Используется переменная окружения HTTP_ACCEPT.
Content-type. Этот заголовок предназначен для идентификации типа передаваемых данных. Наименования типов данных указывается в формате стандарта MIME. Это тот самый формат передачи, который используется методами GET и POST. Сервер никак не интерпретирует рассматриваемый заголовок, а просто передает его сценарию через переменную окружения. Переменная окружения CONTENT_TYPE.
Content-length. Этот заголовок содержит длину передаваемых данных в байтах при использовании метода передачи POST. Переменная окружения CONTENT_LENGTH.

Протокол HTTP

Основные заголовки

Cookie. В этом заголовке хранятся все Cookies. Для установки Cookies используется заголовок Set-Cookie. Переменная окружения HTTP_COOKIE.
Location. Получив этот заголовок вместе с указанным в нем URL, браузер немедленно переходит по указанному URL.
Pragma. Данный заголовок используется для различных целей, одна из которых - это запрет кэширования документа.
Server. Данный заголовок содержит название и версию программного обеспечения сервера.
Referer. Содержит URL страницы, откуда клиент пришел на нашу. Переменная окружения: HTTP_REFERER.
User-Agent. Содержит версию браузера. Переменная: HTTP_USER_AGENT.

Протокол HTTP

Стандарт MIME

MIME (Multipurpose Internet Mail Extensions) — многоцелевые расширения почтового стандарта Интернета. Изначально MIME был создан для указания, какого типа документ вложен в сообщение электронной почты.

MIME-тип задается в виде «тип/подтип». Например: `text/html`

Стандарт MIME определяет семь типов данных:

- application;
- audio;
- image;
- message;
- multipart;
- text
- video;

Протокол HTTP
Коды состояний

В настоящее время выделено пять классов кодов состояния:

- 1xx: Информационный — запрос получен, продолжается обработка.
- 2xx: Успешно — действие было успешно получено, понято и обработано.
- 3xx: Перенаправление — для выполнения запроса должны быть предприняты дальнейшие действия.
- 4xx: Ошибка клиента — запрос имеет плохой синтаксис или не может быть выполнен.
- 5xx: Ошибка сервера — сервер не в состоянии выполнить допустимый запрос.

Протокол HTTP
Переменные окружения

Для связи между web-сервером и приложением используется стандарт CGI (Common Gateway Interface, общий интерфейс шлюза). Эта связь обеспечивается пременными окружения web-сервера, к которым, при необходимости, приложение обращается для получения данных.

- REMOTE_ADDR - IP-адрес хоста, отправляющего запрос
- REMOTE_HOST - Имя хоста, с которого отправлен запрос
- REQUEST_METHOD - Метод, который был использован при отправке запроса
- QUERY_STRING - Информация, находящаяся в URL после знака вопроса
- SCRIPT_NAME - Виртуальный путь к программе, которая должна выполняться

Работа с протоколом HTTP

Cookies

Cookie — это фрагмент информации, который сценарий, при необходимости, сохраняет на клиентской машине. Теоретически использование cookie выглядит следующим образом:

1. Клиент отправляет HTTP-запрос серверу.
2. Сервер отправляет HTTP-ответ, среди прочего включающий в себя заголовок **`Set-Cookie:var=value`**.
3. При необходимости, клиент переходит на другую страницу этого же сервера, путем отправки нового HTTP-запроса, включающего в себя заголовок **`Cookie:var=value`**.
4. Сервер «узнает» клиента и соответствующим образом реагирует на его запрос, если это предусмотрено.

Работа с протоколом HTTP

Cookies. Первый запрос

```
GET / HTTP/1.1 
Host: www.rosnou.ru 
Accept: */* 
Referrer: http://www.google.ru/search?
aq=f&complete=1&hl=ru&newwindow=1&q=%D0%A0%D0%BE%D1%81%D0%9D%D0%9E%
D0%A3&btnG=%D0%9F%D0%BE%D0%B8%D1%81%D0%BA&lr= 
User-Agent: Mozilla 4.0 (compatible; MSIE 6.1,…) 

HTTP/1.1 200 OK 
Date: Sun, 06 Jan 2008 11:19:28 GMT 
Server: Apache/2.2.4 
Pragma: no-cache 
Content-Length: 23341 
Content-Type: text/html; charset=windows-1251 
Set-Cookie: var=value 

<html> . . .
</html>
```

Работа с протоколом HTTP

Cookies. Последующие запросы

```
GET / HTTP/1.1 □
Host: www.rosnou.ru □
Accept: */* □
Referrer: http://www.google.ru/search?
aq=f&complete=1&hl=ru&newwindow=1&q=%D0%A0%D0%BE%D1%81%D0%9D%D0%9E%
D0%A3&btnG=%D0%9F%D0%BE%D0%B8%D1%81%D0%BA&lr= □
User-Agent: Mozilla 4.0 (compatible; MSIE 6.1,…) □
Cookie: var=value □
□
HTTP/1.1 200 OK □
Date: Sun, 06 Jan 2008 11:19:28 GMT □
Server: Apache/2.2.4 □
Pragma: no-cache □
Content-Length: 23341 □
Content-Type: text/html; charset=windows-1251 □
□
<html> . . .
</html>
```

Работа с протоколом HTTP

Cookie. Пример

```php
<?php   // Устанавливаем cookie
    setcookie ("TestCookie", "value");                  //Безвременно
    setcookie ("TestCookie", "value", time()+3600);//На 1 час
    setcookie ("TestCookieArray[1]", "value1");   //Массив Cookie
    setcookie ("TestCookieArray[2]", "value2");
?>
```

Кука станет доступна только после перезагрузки страницы.

```php
<?php   // Читаем cookie
    echo $_COOKIE['TestCookie'];
    echo $_COOKIE['TestCookieArray'][1];
?>
```

```php
<?php   // Удаляем cookie
    setcookie ("TestCookie");     //Устанавливаем куку без значения
    setcookie ("TestCookieArray[1]");
?>
```

Работа с протоколом HTTP
Заголовки ответа

```
HEAD / HTTP/1.1 □
Host: www.rosnou.ru □
Accept: */* □
Referrer: http://www.google.ru/search?
aq=f&complete=1&hl=ru&newwindow=1&q=%D0%A0%D0%BE%D1%81%D0%9D%D0%9E%
D0%A3&btnG=%D0%9F%D0%BE%D0%B8%D1%81%D0%BA&lr= □
User-Agent: Mozilla 4.0 (compatible; MSIE 6.1,…) □
□

HTTP/1.1 200 OK □
Date: Sun, 06 Jan 2008 11:19:28 GMT □
Server: Apache/2.2.4 □
Pragma: no-cache □
Content-Length: 23341 □
Content-Type: text/html; charset=windows-1251 □
□
```

Работа с протоколом HTTP

Заголовок Location

Перенаправляет браузер клиента по указанному в заголовке адресу.

```php
<?php
    if (!headers_sent()) {
        header('Location: http://www.rosnou.ru');
        exit;
    }
?>
<?php
    $host = $_SERVER['HTTP_HOST'];
    $dir  = dirname($_SERVER['PHP_SELF']);
    $page = 'somepage.php';
    if (!headers_sent()) {
        header("Location: http://$host$dir/$page");
        exit;
    }
?>
```

Работа с протоколом HTTP

Заголовок Refresh

Перезагружает страницу по истечении указанного количества секунд.

```php
<?php
    echo "Через 5 секунд страничка обновится!";
    header("Refresh: 5; URL=\"http://rosnou.ru\"");
?>
<?php
    $rate = 5;
    if (!isset($_COOKIE['visits']) || $_COOKIE['visits'] == ""){
        header("Refresh: $rate; URL=\"{$_SERVER['PHP_SELF']}\"");
        setcookie("visits", "1");
        echo "Через $rate секунд страничка обновится 1 раз!";
    }else{
        $visits = $_COOKIE['visits'] + 1;
        header("Refresh: $rate; URL=\"{$_SERVER['PHP_SELF']}\"");
        setcookie("visits", $visits);
        echo "Через $rate секунд страничка обновится $visits раз!";
    }
?>
```

Работа с протоколом HTTP

Заголовок Content-Type

Сообщает браузеру как надо интерпретировать выводимые данные.

```
<?php
    header('Content-type: application/pdf');

    // Этот заголовок форсирует вывод диалога сохранения и
    // рекомендует браузеру имя, с которым надо
    // сохранить файл. В данном случае: downloaded.pdf
    header('Content-Disposition: attachment;
filename="downloaded.pdf"');

    // Читает данные из файла, лежащего на сервере
    readfile('original.pdf');
?>
```

Работа с протоколом HTTP

Заголовоки Cache-Control, Expires, Set-Cookie

Cache-Control используется для управления кэшированием документа.
Expires используется для указания браузерам и транзитным прокси - серверам даты устаревания кэша.
Set-Cookie используется для установки cookie. Функция setcookie() является оберткой этого заголовка.

```
<?php
    // Актуальность кэша исчерпалась 26.07.97
    header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
    // Запрет кэширования
    header("Cache-Control: no-store, no-cache, must-revalidate");
    // Установка cookie name со значением igor
    header("Set-Cookie: name=igor; expires=Wed, 01-Jan-08 14:39:58
GMT");
?>
```

Работа с протоколом HTTP

Базовая аутентификация

```
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
        header('WWW-Authenticate: Basic realm="Entrance"');
        header('HTTP/1.0 401 Unauthorized');
        exit;
  } else
        echo "Привет {$_SERVER['PHP_AUTH_USER']}, ваш пароль
{$_SERVER['PHP_AUTH_PW']}";
?>
<?php
    if(($_SERVER['PHP_AUTH_USER'] == "admin") AND
($_SERVER['PHP_AUTH_PW'] == "megapass"))
        echo "Добро пожаловать!<br>";
    else{
        header('WWW-Authenticate: Basic realm="Entrance"');
        header("HTTP/1.0 401 Unauthorized");
        echo "Вход на страницу закрыт.<br>";
    }
?>
```

Работа с протоколом HTTP
Демонстрация

- Передача формы методом POST;
- Передача формы методом GET;
- Создание, чтение, удаление cookie;
- Перезагрузка страницы;
- Перенаправление;
- Аутентификация.

Работа с протоколом HTTP

Практическая работа

1. Создайте форму с тремя полями: num1, num2, operator. Создайте скрипт калькулятора, который принимает значения, проводит соответствующие вычисления и выводит результат.
2. С помощью cookie создайте механизм запоминания количества посещений страницы пользователем. При каждой загрузке страницы выведите текущее количество посещений.
3. Напишите сценарий, который через каждые 5 секунд будет перегружать страницу. Добавьте вывод текущего времени при каждой перезагрузке.
4. Создайте сценарий базовой аутентификации. Сценарий должен проверять учетные данные и принимать решение о допуске / не допуске пользователя на страницу.