

РНР. Краткое введение.

PHP (*PHP: Hypertext Preprocessor*) –

скриптовый язык программирования,
созданный для генерации HTML-страниц
на стороне веб-сервера.

Основные стандартные типы данных:

- Целый тип (integer)
- Вещественный тип (float, double и real – одно и то же в PHP)
- Логический тип (boolean)
- Строковый тип (string)
- Специальный тип NULL
- Ассоциативные массивы(array)

PHP является языком программирования с динамической типизацией(не требует объявления типа переменных/самих переменных)

Синтаксис.

Пример:

```
1 PHP код по умолчанию размещается в файлах с расширениями .php
2 и .html</br>
3 <?
4     // Интерпретатор выполняет всё, что находится между
5     // <? .. ? > или <?php .. ? >
6     echo 'Hello <u>World!</u></br>';
7     /*
8         Команда 'echo' не является функцией, поэтому
9         строка выше не обрамлена круглыми скобками.
10    */
11 ?>
12 При этом всё, что находится вне указанных тегов - <b>НИКАК</b> не
13 обрабатывается парсером и может содержать валидный HTML код,
14 адекватно воспринимаемый браузером. </br>
15 // Даже то, что считается комментарием в PHP
16 <!-- Обычный HTML-комментарий не будет виден в браузере -->
```

Синтаксис.

Результат:

Nightly

PHP код по умолчанию размещается в файлах с расширениями .php и .html

Hello World!

При этом всё, что находится вне указанных тегов - **НИКАК** не обрабатывается парсером и может содержать валидный HTML код, адекватно воспринимаемый браузером.

// Даже то, что считается комментарием в PHP

Синтаксис.

Пример:

```
1 <?php
2     $a = 99;
3     $b = (real)1.1;
4
5     $first_str = 'Сумма значений переменных $a и $b(';
6     $second_str = "а именно, результат выражения $a + $b): ";
7
8     // Обратите внимание на . (точку) - это оператор склейки строк
9     echo $first_str . $second_str . ($a + $b) . '</br>';
10
11    // Стандартные операции над числами: -, /, %, ++, --
12    // А так же побитовые операции: and(&&), or(||), xor, !
13    // Операции сравнения: >, <, >=, <=, !=(<>), ==, !=, ===, !==
14
15    echo '<i>Определение типа переменных.</i></br>';
16    echo '$a имеет тип ' . gettype($a) . '</br>';
17    echo '$b имеет тип ' . gettype($b) . '</br>';
18 ?>
```

Синтаксис.

Результат:

Nightly

Сумма значений переменных \$a и \$b(а именно, результат выражения $99 + 1.1$): 100.1

Определение типа переменных.

\$a имеет тип integer

\$b имеет тип double

Синтаксис.

Пример:

```
1 <?
2 function reverse_string($text) {
3     if (!is_string($text))
4         die('Ошибка: неверный тип аргумента.');
```

5

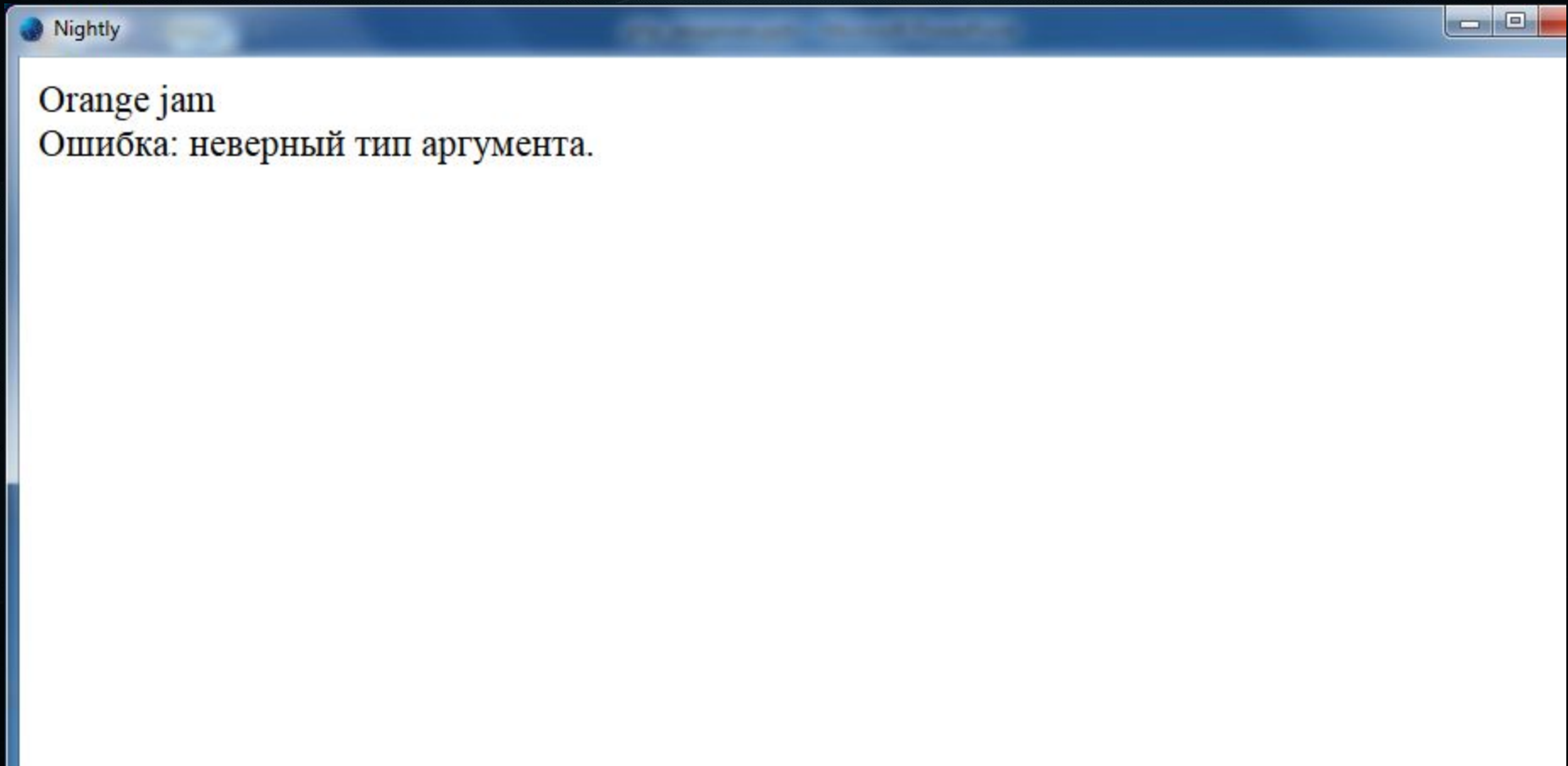
```
6     $len = strlen($text);
7
8     for($i = 0; $i < $len / 2; $i++) {
9         $tmp = $text[$i];
10        $text[$i] = $text[$len - $i];
11        $text[$len - $i] = $tmp;
12    }
13
14    return $text;
15 }
```

16

```
17 echo reverse_string('maj egnarO') . '</br>';
18 echo reverse_string(100) . '</br>';
19 ?>
```


Синтаксис.

Результат:



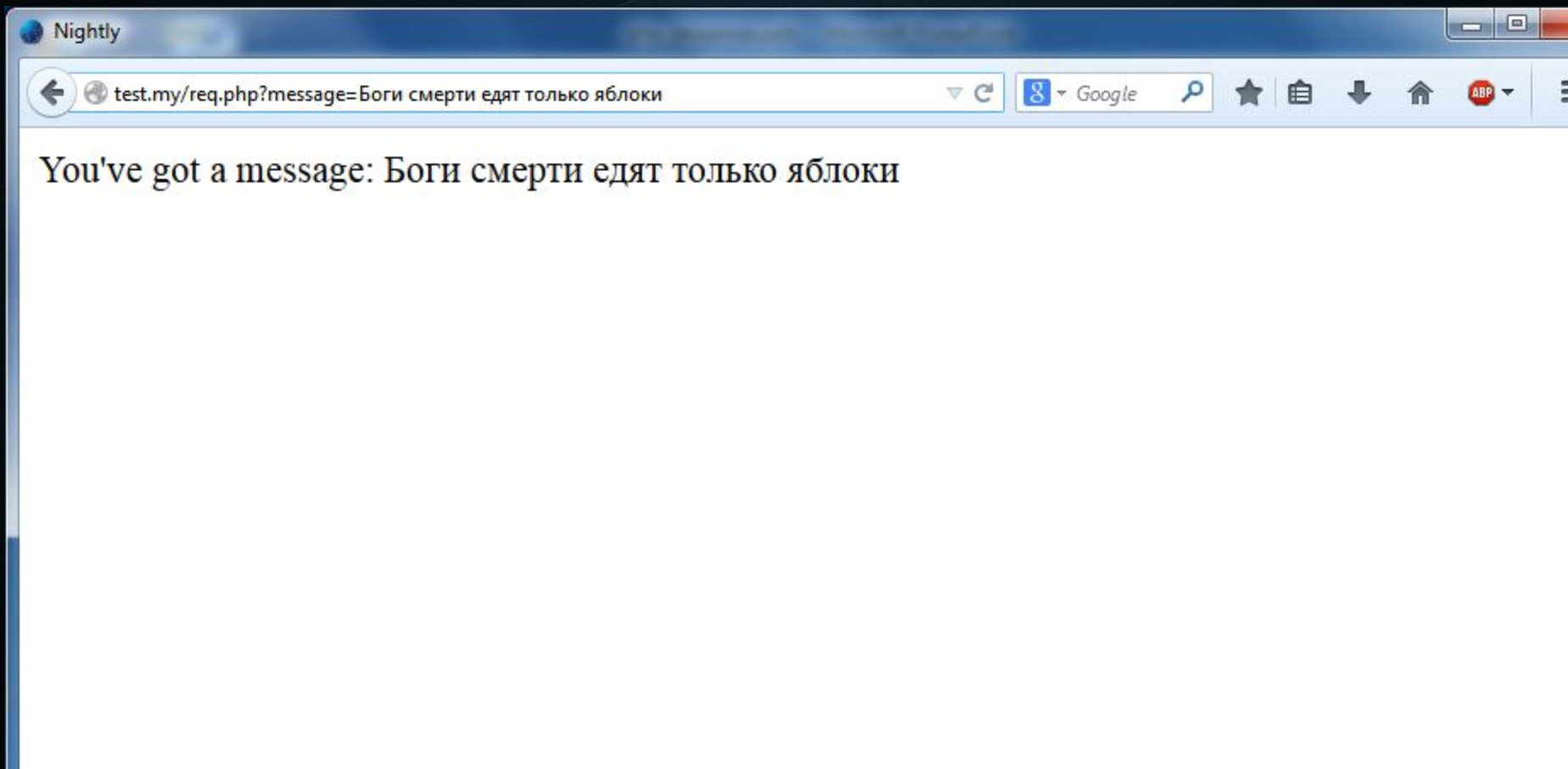
Обработка запросов.

Пример:

```
1  <?
2  /*
3     Суперглобальный массив $_GET содержит все параметры,
4     переданные скрипту методом GET. Так же для приёма данных
5     существуют суперглобальные массивы $_POST(содержит
6     данные, присланные методом POST), и $_REQUEST(сюда
7     складываются ВСЕ запросы, присланные обоими методами).
8  */
9
10 if ( isset( $_GET['message'] ) ) {
11     $message = $_REQUEST['message'];
12     echo "You've got a message: $message</br>";
13 } else
14     echo 'No messages for you</br>';
15 ?>
```

Обработка запросов.

Результат:



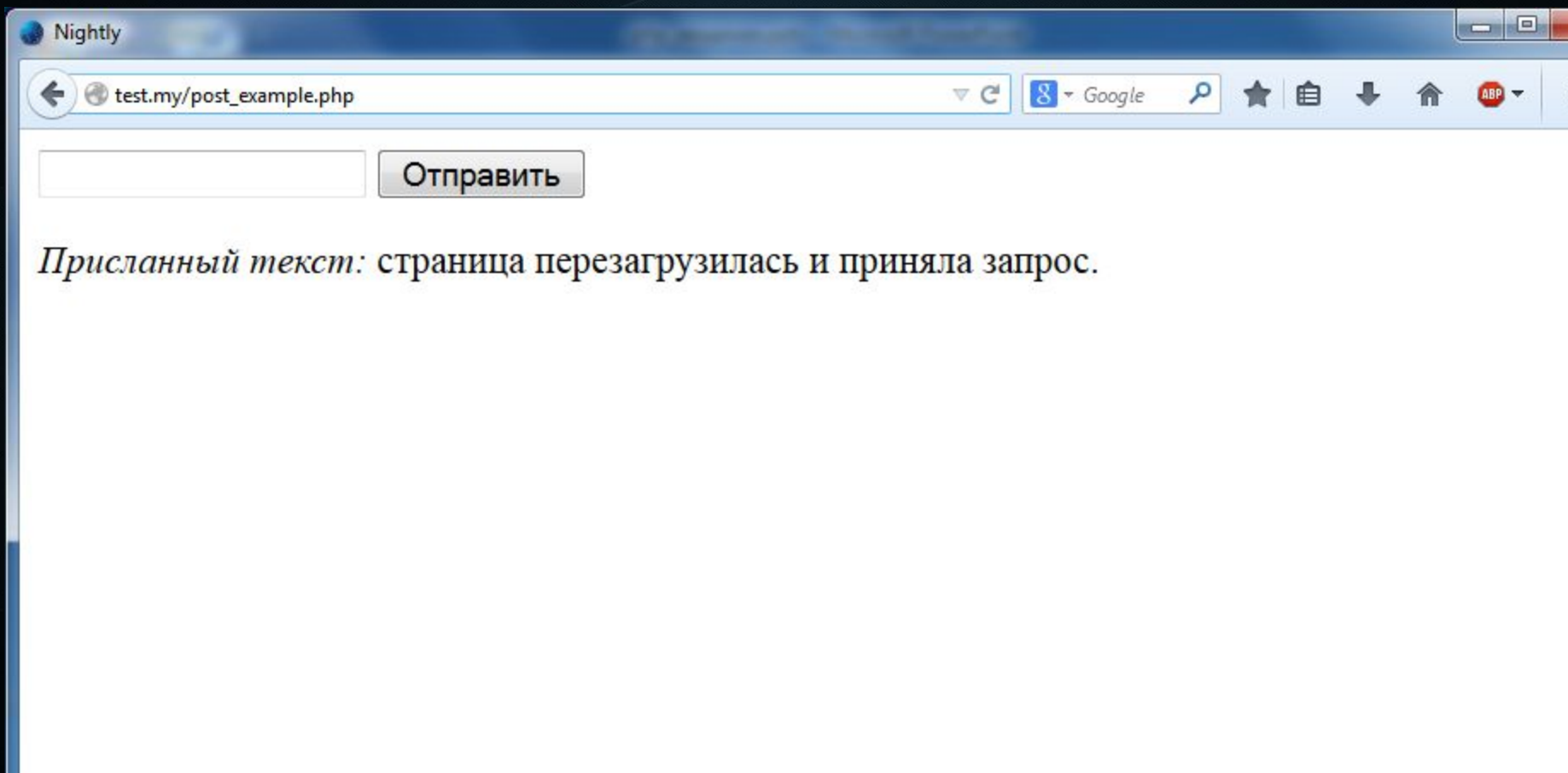
Обработка запросов.

Пример:

```
1 <!-- Простейшая HTML-форма, отправляющая POST-запрос. -->
2 <form action="post_example.php" method="post" accept-charset="
  utf-8">
3   <input type="text" name="text" value="">
4   <input type="submit" name="ok" value="Отправить">
5 </form>
6 <?
7   if (isset($_REQUEST['text']) && $_REQUEST['text'] != '') {
8     $text = $_REQUEST['text'];
9     echo "<i>Присланный текст:</i> $text";
10  }
11 ?>
```

Обработка запросов.

Результат:



Какому методу отдать предпочтение?

- Если нужен человекочитаемый URL(например, для SEO оптимизации) – выбираем GET.
- Нужно передавать данные, которые не должны быть видимы пользователю (например, логин/пароль) и не должны палиться в логах сервака – выбираем POST.
- Передаётся значительный объём данных или к форме прикреплен файл – выбираем метод POST.

Request-Response.

Данная модель предполагает обмен клиент-сервер сообщениями вида запрос(клиент)-ответ(сервер).

Сообщение состоит из трёх частей: стартовая строка, заголовки, тело. При этом обязательна только стартовая строка.

`METHOD URI HTTP/VERSION` – так выглядит стартовая строка для запроса. Здесь `METHOD` – имя метода (`GET`, `POST` или другие), `URI` – идентификатор ресурса, `VERSION` – используемая версия протокола HTTP

Заголовки – набор пар «имя:значение». Они содержат служебную информацию вроде используемой на странице кодировки и т. д.

Тело содержит передаваемые данные(в случае запроса), и, как правило, содержимое HTML-страницы(в случае ответа).

Request-Response.

Пример запроса:

GET /index.php HTTP/1.1

Host: example.com

User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru;
rv: 1.9b5)Gecko/2008050509 Firefox/3.0b5

Accept: text/html

Connection: close

Request-Response.

Пример ответа:

HTTP/1.0 200 OK

Server: nginx/0.6.31

Content-Language: ru

Content-Type: text/html; charset=utf-8

Content-Length: 1234

Connection: close

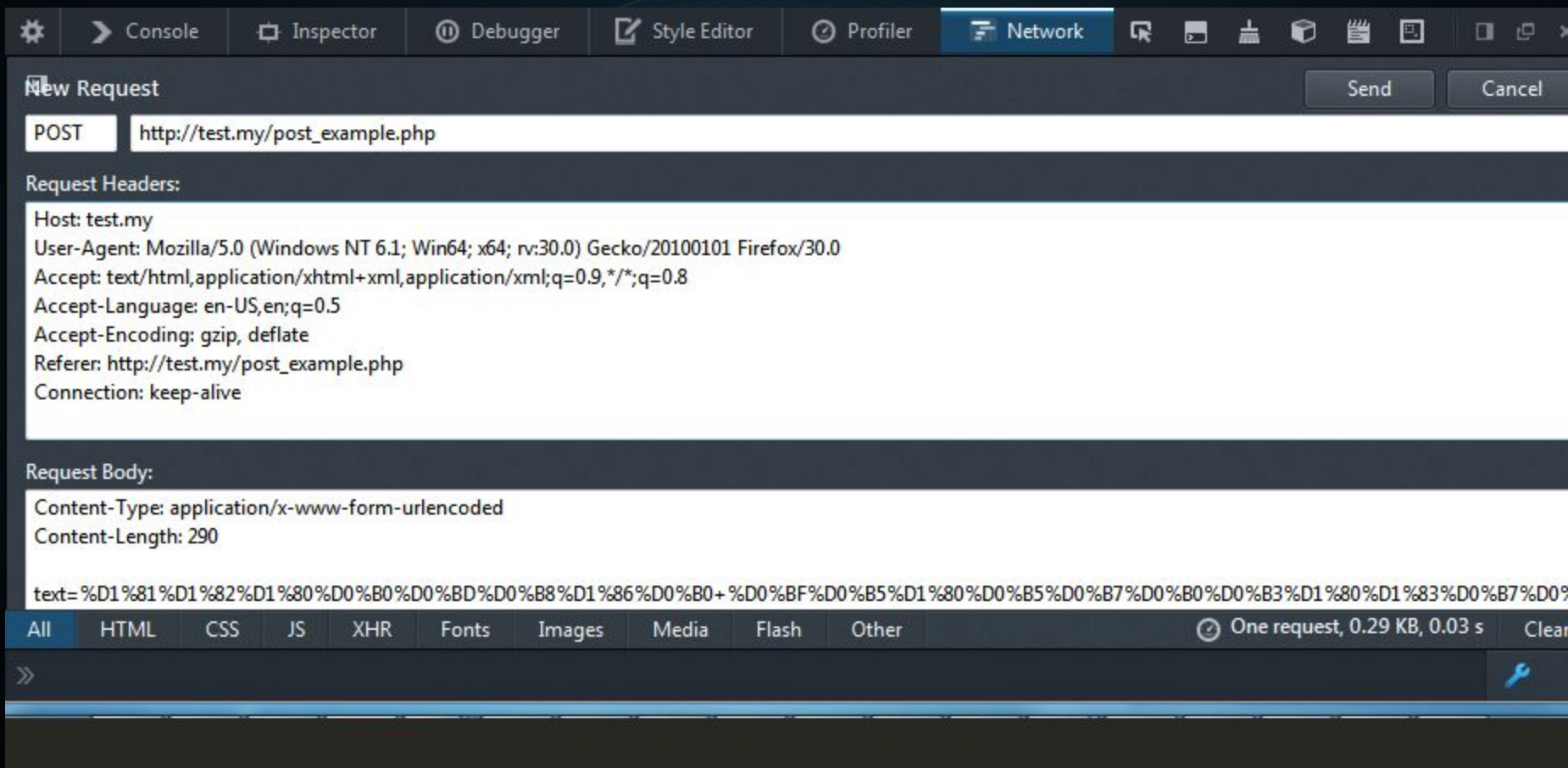
<html>

.... Какой-то html-код

</html>

Обработка запросов.

Заголовок и тело запроса как есть:



The screenshot shows the 'New Request' dialog in a browser's developer tools. The request is a POST to `http://test.my/post_example.php`. The headers and body are as follows:

Request Headers:

- Host: test.my
- User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:30.0) Gecko/20100101 Firefox/30.0
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
- Accept-Language: en-US,en;q=0.5
- Accept-Encoding: gzip, deflate
- Referer: http://test.my/post_example.php
- Connection: keep-alive

Request Body:

- Content-Type: application/x-www-form-urlencoded
- Content-Length: 290

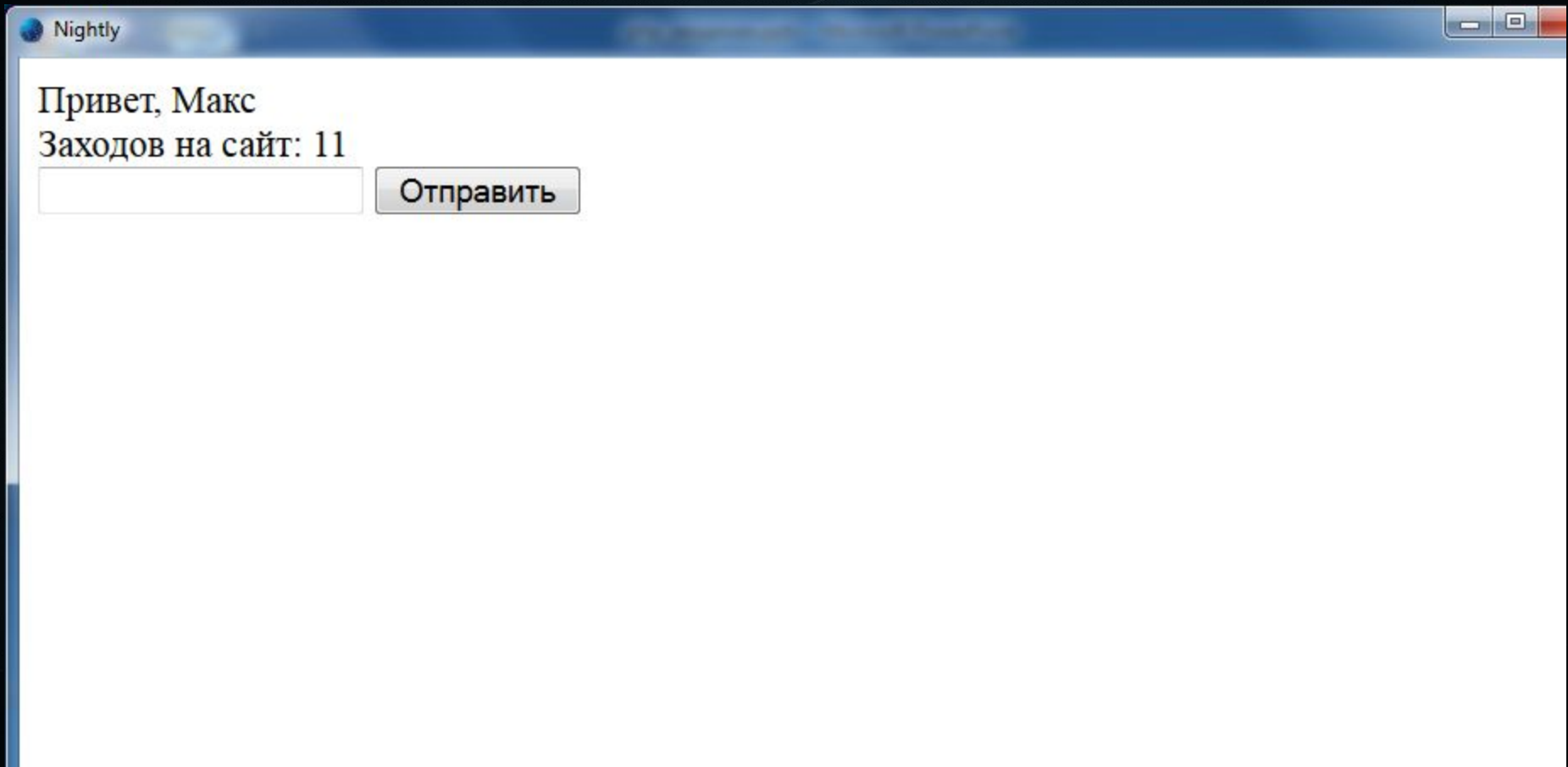
The body content is a URL-encoded string: `text= %D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0+ %D0%BF%D0%B5%D1%80%D0%B5%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%`

The bottom of the dialog shows a summary: 'One request, 0.29 KB, 0.03 s'.

Cookies.

```
1  <?
2  // ! Установка кук должна происходить ДО осуществления вывода
   // посредством echo или как-либо иначе
3  if (isset($_REQUEST['username']) && $_REQUEST['username'] != '
   ') {
4      $username = $_REQUEST['username'];
5      setcookie("username", $username, time() + 3600);
6
7      if (!isset($_COOKIE['count']))
8          setcookie("count", 0, time() + 3600);
9      else
10         setcookie("count", $_COOKIE['count'] + 1, time() +
            3600);
11     }
12     // После того, как куки установлены(при необходимости), можем
   // вывести приветствие
13     if (isset($_COOKIE['username'])) {
14         echo 'Привет, ' . $_COOKIE['username'] . '</br>';
15
16         if (isset($_COOKIE['count']))
17             echo 'Заходов на сайт: ' . $_COOKIE['count'] . '</br>';
18     }
19     ?>
20     <form action="cookies.php" method="post" accept-charset="utf-8">
21         <input type="text" name="username" value="">
22         <input type="submit" name="ok" value="Отправить">
23     </form>
```

Cookies.



Спасибо за внимание!

