

## Урок №4

### Запросы HTTP, параметры URL и формы HTML.

В этом уроке вы

- узнаете, какие бывают типы запросов HTTP;
- узнаете, как формируются параметры запроса;
- научитесь передавать параметры через URL;
- научитесь обрабатывать параметры, передаваемые через URL;
- научитесь обрабатывать отправку HTML форм.

#### 1. Типы запросов HTTP

Итак, мы наконец-то подошли к уроку, в котором всё начинается по-настоящему. Наконец-то прояснится, каким образом и какие выгоды даёт PHP.

Для начала немного вспомним материал первого урока, а именно, что такое HTTP.

**HTTP** (сокр. от англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — это протокол передачи данных между сервером и клиентом.

Как мы уже знаем, существуют HTTP-запросы (передача данных от клиента серверу) и HTTP-ответы (передача данных от сервера клиенту). Сегодня нас будут интересовать только запросы. Они бывают двух типов.

1. GET
2. POST

GET используется, при наборе адреса сайта в строке браузера или перехода по ссылке. POST служит для отправки форм, например, при регистрации на сайте, при публикации комментария к статье. Для отправки формы обычно нужно нажать кнопку "Отправить" или наподобие того.

У каждого из типов запроса есть своё логическое назначение. Параметры, передаваемые через GET, служат для чтения информации из Интернета. Например, мы находимся на странице просмотра новостей, и серверу необходимо знать номер новости, которую нам нужно отобразить. Этот номер передаётся именно методом GET.

Параметры, передаваемые через POST, служат для публикации информации на сайтах. Например, администратор сайта заполняет форму добавления новой статьи. То, что он набирает в полях ввода, будет передано на сервер именно методом POST.

По сути, назначение способов передачи выглядит следующим образом:

- 1) GET – читаем информацию из Интернета
- 2) POST – публикуем информацию в Интернет

## 1. URL и параметры запроса

Снова обратимся к первому уроку. **URL** (англ. Uniform Resource Locator) – единообразный локатор (определитель местонахождения) ресурса. URL — это стандартизированный способ записи адреса ресурса в сети Интернет.

Проще говоря, URL - это адрес страницы в Интернет. Однако, на самом деле через URL может передаваться не только сам адрес страницы, но и дополнительные параметры для неё.

Если параметры не передаются, то URL имеет привычный нам вид:

```
http://<хост>/<путь>
```

Например:

```
http://site.ru/news.php
```

В случае передачи параметров, они добавляются к URL-адресу после вопросительного знака:

```
http://<хост>/<путь>?<параметры>
```

где <параметры> - это набор пар вида:

<имя>=<значение>

разделенных символом &.

Например:

```
http://site.ru/news.php?id=15&show_comments=yes
```

В данном запросе скрипту передано 2 параметра: id=15 и show\_comments=yes. И самое главное сейчас - понять, зачем они вообще могут быть нужны скрипту.

Динамическая страница, в отличие от статической, может формировать для клиента различные ответы, зависящие от определённых условий. Параметры, как раз, такими условиями и являются.

Рассмотри это на примере странички news.php. По сути, она может представлять собой просто шаблон из html-кода, в котором оставлено место под текст статьи. Параметр id=15 показывает РНР-скрипту, что необходимо сделать запрос к базе данных и достать оттуда именно 15-ую новость, а параметр show\_comments=yes показывает, что нужно также получить и отобразить комментарии к данной записи.

В этом и заключается суть РНР. Ведь если бы мы делали такой сайт только с помощью чистого HTML-кода, то нам пришлось бы на каждую новую статью создавать отдельную страничку. В итоге это привело бы к огромному количеству однотипных документов на нашем сайте, в которых, к тому же, было бы много повторяющегося кода (описание частей, которые есть на каждой странице: шапка, подвал и т.п.).

## 2. Обработка параметров URL

Только что мы посмотрели, каким образом можно передавать параметры через URL-адрес. Теперь пришло время научиться их обрабатывать. Для этого создадим небольшую галерею фотографий.

У нее будет две страницы:

1. отображает ссылки на картинки;
2. показывает изображение полностью.

Для начала сделайте следующую структуру файлов и папок:



Теперь скопируйте в папку `img` 4 картинки и переименуйте их таким образом, чтобы названия были цифрами, которые идут по порядку, например:



Рабочее пространство подготовлено, теперь можно писать код. Начнём с `photo.php`:

```
<?php
    $id = $_GET['id']; // Считываем передаваемый параметр
?>
<html>
    <head>
        <title>Просмотр картинки № <?php echo $id;?></title>
    </head>
    <body>
        
        <h2>Какое-то описание картинки</h2>
    </body>
</html>
```

Подробно разберем, что же мы написали. По задаче, на данной странице нам необходимо отобразить определённую картинку. Чтобы понять, какую именно картинку показывать, нужно считать определённый параметр – номер изображения – из URL-адреса.

Параметры, переданные через URL, хранятся в системной переменной `$_GET`. Она представляет собой ассоциативный массив (или, по-другому, словарь).

Ассоциативный массив (или словарь) - это такая структура данных, которая содержит пары ключ-значение.

Ключи словаря `$_GET` - это имена параметров. Мы считаем, что пользователь будет обращаться к странице по следующему адресу:

```
http://<Ваш сайт>/photo.php?id=<какой-то номер>
```

Это означает, что в массиве `$_GET` будет содержаться один элемент – по ключу `id` будет лежать переданный номер, который нам и нужен. Также обратите внимание на то, что массив `$_GET` автоматически формируется PHP-интерпретатором, а мы просто пользуемся теми данными, которые в нём есть.

Во второй строчке `photo.php` мы считываем номер картинки и затем подставляем его в статический HTML-код. В итоге, теперь, если пользователь обратится по адресу

```
http://<Ваш сайт>/photo.php?id=1
```

то увидит первую картинку, по адресу

```
http://<Ваш сайт>/photo.php?id=2
```

- вторую и т.д.

Однако, вряд ли пользователю понравится самому вручную менять адрес сайта. Поэтому на главной страничке мы должны сделать ссылки на все изображения. Создадим `index.php`:

```
<html>
  <head>
    <title>Галерея изображений</title>
  </head>
  <body>
    <h2>Какое-то описание картинки</h2>
    <?php
      for($i = 1; $i <= 4; $i++)
        echo "<a href = photo.php?id=$i >Картинка №$i</a><br/>";
    ?>
  </body>
</html>
```

И снова разберёмся с тем, что мы написали.

Необходимо было создать 4 ссылки (столько же, сколько всего у нас картинок). Вручную это было бы грустно делать (представьте, что картинок станет штук 50), и поэтому мы воспользовались циклом. На каждой его итерации на экран выводится ссылка вида:

```
<a href = photo.php?id=[номер картинки] >Картинка №=[номер картинки]</a><br/>
```

где [номер картинки] – число от 1 до 4.

В итоге, при нажатии на одну из этих ссылок, пользователь попадёт на страницу `photo.php`, которой будет передан дополнительный параметр методом GET, а именно, номер картинки.

Данная галерея изображений состоит из очень небольшого количества строк кода, но даёт огромное преимущество перед реализацией на чистом HTML в том случае, если у нас будет много картинок.

### 3. Обработка отправки HTML формы

В предыдущем примере мы отправляли запросы методом GET. Теперь познакомимся ближе с методом POST и формами HTML. Для этого создадим скрипт, который будет складывать числа.

Создайте файл `sum.php`:

```
<?php
    if(isset($_POST['a']) && isset($_POST['b']))
        $result = $_POST['a'] + $_POST['b'];
    else
        $result = "";
?>
<html>
    <head>
        <title>Галерея изображений</title>
    </head>
    <body>
        <form method="post">
            <input type="text" name="a" />
            +
            <input type="text" name="b" />
            <input type="submit" value="=" />
            <?php echo $result; ?>
        </form>
    </body>
</html>
```

Для начала пропустим весь PHP-код и просто разберёмся с HTML-формой. У тега `form` мы указали метод передачи данных. Это обязательно нужно делать, иначе параметры будут передаваться через URL-адрес.

Тег `input` бывает различных типов (определяется атрибутом `type`). Нам нужно два текстовых поля (`type="text"`) и одна кнопка (`type="submit"`). Текстовым полям мы дали имена "a" и "b", чтобы из скрипта получить введенные туда пользователем значения.

По аналогии с методом GET, параметры, переданные методом POST, содержатся в системной переменной `$_POST`. Это также, как и `$_GET`, ассоциативный массив. Ключами в массиве `$_POST` являются значения атрибутов `name` у элементов на форме. Соответственно, в нашем случае в массиве будет два элемента с ключами `a` и `b`.

Единственный важный момент, который немного усложняет логику скрипта, заключается в том, что на данную страничку пользователь может попасть как методом POST, так и методом GET. В том случае, если он нажал на кнопку отправить, имеем метод POST, так как мы указали это на форме в атрибуте `method`. Во всех остальных вариантах (набрал адрес в строке браузера, перешёл по ссылке) пользователь попадает на страницу методом GET.

Поэтому, в самом начале скрипта `index.php` мы проверяем, каким методом совершён запрос к странице. Для этого проверяем наличие элементов "a" и "b" в массиве `$_POST` с

помощью специальной функции `isset()`. Она возвращает `true` в том случае, если элемент существует, и `false` - в противном.

Если мы определили, что запрос осуществлялся методом `POST`, складываем значения, которые передал нам пользователь. Если же проверка вернула на `false` – просто запоминаем в переменную `$result` пустую строку, так как пользователь ещё не пытался произвести вычислений.

Здесь выгода `PHP` очевидна, так как без него мы просто не смогли бы обработать отправку формы.

## Резюме

В этом уроке мы, наконец, перешли от скучной теории к долгожданной практике! Теперь вы можете писать боевые скрипты, обрабатывающие настоящие `HTTP` запросы.

Самое главное сейчас – понять и прочувствовать, какую выгоду даёт `PHP`. Очень важно запомнить, что существуют только `GET` и `POST`. Дело в том, что, в основном, логика работы любой странички, как простой, так и сложной, зависит, в первую очередь, именно от того, каким методом был сделан запрос.

## Самоконтроль

- ✓ Какие два типа запросов Вы знаете
- ✓ Чем по смыслу отличаются `GET` и `POST`
- ✓ В каком из этих методов параметры передаются через `URL`-адрес
- ✓ Каким образом параметры передаются через `URL`-адрес
- ✓ В каком массиве хранятся параметры, переданные через `URL`-адрес
- ✓ Как передать через `URL`-адрес больше одного параметра
- ✓ Какую выгоду даёт возможность передачи параметров через `URL`-адрес
- ✓ Какие основные теги `HTML`-форм Вы знаете?
- ✓ Зачем на форме писать `method="post"`
- ✓ Каким образом формируются ключи в массиве `$_POST`
- ✓ Что делает функция `isset`
- ✓ В каком случае пользователь попадает на страницу методом `POST`
- ✓ В каком случае пользователь попадает на страницу методом `GET`

## Домашнее задание

### Базовый блок.

1. Обязательно сделайте скрипты, приведенные в качестве примеров в этом уроке.
2. Превратите получившийся сумматор в калькулятор с четырьмя операциями: сложение, вычитание, умножение, деление. Не забудьте обработать деление на ноль!

Выбор операции можно осуществлять с помощью тега `<select>`.



The image shows a basic calculator interface. It consists of two input fields for numbers, a dropdown menu for selecting an operation (currently showing '+'), and an equals button (=). The dropdown menu is open, showing options for '+', '-', '\*', and '/'. The '+' option is currently selected.

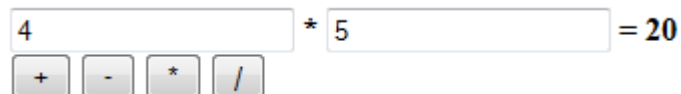
### Продвинутый блок.

3. Создайте калькулятор, который будет определять тип выбранной пользователем операции, ориентируясь на нажатую кнопку.



The image shows a calculator interface. It has two input fields for numbers, an equals button (=), and four buttons for operations: '+', '-', '\*', and '/'. The input fields are currently empty.

Данные, введённые пользователем в поля, должны сохраняться и выводиться вместе с результатом вычисления.



The image shows a calculator interface. The first input field contains the number '4', the second input field contains the number '5', and the result '20' is displayed to the right of the second field. The operation buttons '+', '-', '\*', and '/' are visible below the input fields.