#### МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**WEB-ПРОГРАММИРОВАНИЕ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №1

Выполнил Учащийся группы ПО-455

И. А. Дудкин

Шифр 16

2023**27 Опишите структуру PHP-сценария, основные элементы синтаксиса, способы внедрения PHP-сценария в веб-документ, назначение протокола HTTP и его методы, особенности передачи данных формы PHP-сценарию.**

В ходе своей работы модуль *PHP* читает и исполняет сценарий (*скрипт*) – текстовый файл, содержащий набор команд. Результат выполнения сценария – это *HTML*-документ, который *PHP* передает серверу *APACHE*, а тот уже – запрашивающему клиентскому браузеру.

*PHP*-сценарий следует сохранять в файлах с расширением имени*.php* в каталоге *htdocs* сервера *APACHE*. Запускать сценарий на выполнение можно только из браузера (*недвойным щелчком!),*набрав в адресной строке:

HTTP://localhost/имя.php

В этом случае мы создаем запрос по протоколу *HTTP*. В этом случае *APACHE* передает *PHP*-сценарий на обработку модулю *PHP*. Модуль *PHP* будет обрабатывать и выполнять только те команды скрипта, которые заключены в специальные теги-*дескрипторы PHP*:

*<?php*

*код на PHP*

*?>*

Действия, которые должен выполнить *PHP*-модуль, указываются *PHP*-операторами, помещающимися между открывающими и закрывающими скобками дескриптора.

*PHP*-сценарий можно встраивать в *HTML*-документ. В этом случае модуль *PHP* передает веб-серверу теги *HTML* без изменений и начнет обработку только команд, помещенных между дескрипторами *PHP*:

<HTML>

<HEAD>

<TITLE>

Сайт с PHP-кодом

</title>

</head>

<BODY BGCOLOR=silver TEXT=blue>

<?php

Echo ‘Это php-скрипт’;

/>

</body>

</html>

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

Методы

Метод HTTP (англ. HTTP Method) — последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово, записанное заглавными буквами. Обратите внимание, что название метода чувствительно к регистру.

Сервер может использовать любые методы, не существует обязательных методов для сервера или клиента. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус 501 (Not Implemented). Если серверу метод известен, но он неприменим к конкретному ресурсу, то возвращается сообщение с кодом 405 (Method Not Allowed). В обоих случаях серверу следует включить в сообщение ответа заголовок Allow со списком поддерживаемых методов.

Кроме методов GET и HEAD, часто применяется метод POST.

OPTIONS

Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.

Предполагается, что запрос клиента может содержать тело сообщения для указания интересующих его сведений. Формат тела и порядок работы с ним в настоящий момент не определён; сервер пока должен его игнорировать. Аналогичная ситуация и с телом в ответе сервера.

Для того, чтобы узнать возможности всего сервера, клиент должен указать в URI звёздочку — «\*». Запросы «OPTIONS \* HTTP/1.1» могут также применяться для проверки работоспособности сервера (аналогично «пингованию») и тестирования на предмет поддержки сервером протокола HTTP версии 1.1.

Результат выполнения этого метода не кэшируется.

GET

Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»:

GET /path/resource?param1=value1&param2=value2 HTTP/1.1

Согласно стандарту HTTP, запросы типа GET считаются идемпотентными[2]

Кроме обычного метода GET, различают ещё

Условный GET — содержит заголовки If-Modified-Since, If-Match, If-Range и подобные;

Частичный GET — содержит в запросе Range.

Порядок выполнения подобных запросов определён стандартами отдельно.

HEAD

Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше — копия ресурса помечается как устаревшая.

POST

Основная статья: POST (HTTP)

Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

В отличие от метода GET, метод POST не считается идемпотентным[2], то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты (например, после каждой отправки комментария будет появляться очередная копия этого комментария).

При результате выполнения 200 (Ok) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием URI нового ресурса в заголовке Location.

Сообщение ответа сервера на выполнение метода POST не кэшируется. POST http://soglasiepre.virtusystems.ru/Authentication\_JSON\_AppService.axd/Login HTTP/1.1 Accept: \*/\* User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; MathPlayer 2.10d; MRA 5.4 (build 02647); .NET CLR 2.0.50727; Content-Type: application/json; charset=utf-8 Host: vfos.virtusystems.ru Content-Length: 76 Expect: 100-continue Connection: Keep-Alive

PUT

Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существует ресурса, то сервер создаёт его и возвращает статус 201 (Created). Если же ресурс был изменён, то сервер возвращает 200 (Ok) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки Content-\*, передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или недопустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented).

Фундаментальное различие методов POST и PUT заключается в понимании предназначений URI ресурсов. Метод POST предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя PUT, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.

Сообщения ответов сервера на метод PUT не кэшируются.

PATCH

Аналогично PUT, но применяется только к фрагменту ресурса.

DELETE

Удаляет указанный ресурс.

TRACE

Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.

CONNECT

Преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищённого SSL-соединения через нешифрованный прокси.

HTTP-методы GET и POST — самые распространённые способы отправить или получить данные с сервера. Но в разных случаях оба метода могут быть небезопасными или неудобными в использовании. В этой заметке рассмотрим, какой метод когда использовать.

GET — метод для чтения данных с сайта. Например, для доступа к указанной странице. Он говорит серверу, что клиент хочет прочитать указанный документ. На практике этот метод используется чаще всего, например, в интернет-магазинах на странице каталога. Фильтры, которые выбирает пользователь, передаются через метод GET.

:authority: htmlacademy.ru

:method: GET

:path: /tutorial/php/http-header

POST — метод для отправки данных на сайт. Чаще всего с помощью метода POST передаются формы.

URL-адрес запроса: https://htmlacademy.ru/consulting

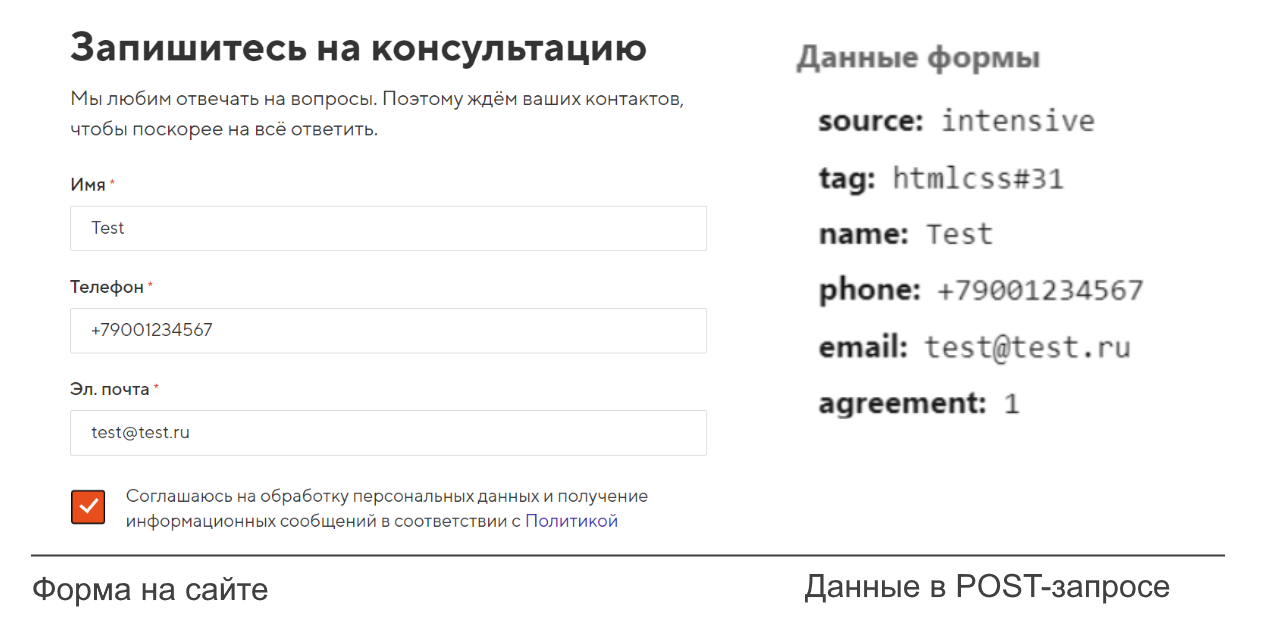
Метод запроса: POST

Код состояния: 200

## Формат запроса

Протокол HTTP очень прост и состоит, по сути, из двух частей — заголовков и тела запроса или ответа.

**Тело запроса** — это информация, которую передал браузер при запросе страницы. Но тело запроса присутствует только если браузер запросил страницу методом POST. Например, если отправлена форма, то телом запроса будет содержание формы.



Пример GET-запроса. Информация передаётся прямо в заголовке.

GET /blog/?name1=value1&name2=value2 HTTP/1.1

Host: htmlacademy.ru

Пример POST-запроса. Информация передаётся в теле запроса:

POST /blog/ HTTP/1.1

Host: htmlacademy.ru

name1=value1&name2=value2

## GET для безопасных действий, POST для опасных

Говоря совсем просто, GET-запросы лучше не использовать с приватной информацией. Вот почему:

* **Они кэшируются.** Это значит, что логин и пароль, переданные через GET-запрос, могут остаться в интернете навсегда, например, в веб-архиве или кэше Гугла.
* **Остаются в истории браузера.** Чтобы узнать, какие данные отправлялись, достаточно нажать Ctrl+H.
* **Сохраняются в закладках и пересылаются.** Можно не обратить внимания и опубликовать в соцсетях или отправить ссылку с приватной информацией в GET-запросе.
* **Сохраняются в логах сервера.** Например, нельзя отправлять данные банковских карт через GET-запрос, так как это создаёт риски для пользователей.

Таким образом, любые важные данные — логины, пароли, данные карты, персональные данные — лучше передавать с помощью метода POST. Также метод POST поддерживает тип кодирования данных multipart/form-data, что позволяет передавать файлы.

## Ещё раз коротко

**GET**

* Фильтры в интернет-магазинах
* Передача параметров через ссылку
* Другие безопасные запросы

**POST**

* Любые формы с паролями или банковскими картами
* Формы заявок с персональными данными
* Отправка файлов

**57 Дан массив a = [15, 97, 2, 158, 26, 268, 88, 105, 43]. При помощи цикла while найти сумму нечетных элементов данного массива.**

**61 Дан двумерный массив (матрица), состоящий из N строк и N элементов, где N вводится с клавиатуры при помощи prompt. Заполните каждую строку случайными числами в диапазоне от -10 до 30. Определите:**

**- количество строк, не содержащих ни одного нулевого элемента;**

**- максимальное из чисел в заданном массиве.**

**97 Спроектировать структуру базы данных о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.**

**Запросы:**

**а) вывести информацию о квартирах, расположенных на заданном этаже;**

**б) вывести информацию о квартирах, имеющих площадь из указанного интервала.**

**Список использованных источников**

1 Минник, К. JavaScript / К. Минник, Е. Холланд. - М. : Москва, 2017. - 320 с.

2 Маккоу, А Веб-приложения на JavaScript / А. Маккоу. - СПб. : Питер, 2014. - 285 с.

3 Макфарланд, Д. JavaScript и jQuery. Исчерпывающее руководство / Д. Макфарланд. - СПб.: Питер, 2016. - 880 с.

4 Жадаев, А.Г. PHP для начинающих / А.Г. Жадаев – СПб.: Питер, 2014. – 288 с.

5 Котеров Д.В. PHP 7 в подлиннике / Д.В. Котеров – СПб.: Питер, 2016. – 1073 с.

6 Никольский А.П. JavaScript на примерах / А.П. Никольский – СПб.: Питер, 2018. – 272 c.