1) Основные понятия. Структура URL

Аббревиатура URL расшифровывается как Uniform Resource Locator. В переводе это значит «единый указатель ресурсов». Детище Тима Бернеса-Ли, «отца» всемирной паутины, создан для упрощения структурирования данных в сети Интернет и приведения всех адресов к единому виду.

Принятое произношение: «у-эр-эл» или «ю-ар-эл». В разговорной речи допустимо произносить эту аббревиатуру как «УРЛ». В этой статье мы рассмотрим что такое URL, какова его структура и технические особенности которые необходимо знать любому веб-мастеру.

**Схема и структура URL адресов**

Что же такое URL? Иными словами, это путь до какого-то файла: HTML, видео, аудио, текстового и пр. Этим адресом обладают все документы, находящиеся в сети Интернет. Только с помощью URL сервер может обеспечить доступ к своим файлам для любого пользователя всемирной паутины.

Разумеется, существует стандарт. Он-то и определяет, как именно должен выглядеть URL к тому или иному документу. Этот стандарт используется не только для доступа к сайтам и файлам, по протоколу http и https, его же используют и при передаче файлов на сервер по протоколу ftp и в других протоколах. Но нас они в данном случае не интересуют.

**Как указать url адрес к веб-ресурсам?**

Вообще URL может содержать множество параметров. Но для простого обращения к определенному файлу лежащему в какой то папке вашего сайта необходимо задействовать лишь некоторые из них. Поэтому, например, чтобы получить содержимое определенного файла размещенного на данном сайте достаточно указать следующее:

http://webmastermix.ru/direktoriya/fail.html

Давайте рассмотрим все подробно:

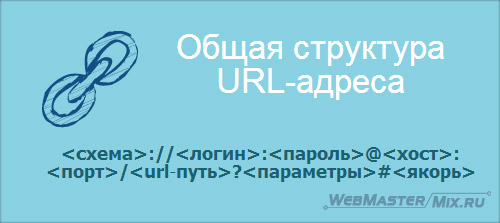
* http:// - протокол передачи данных.
* webmastermix.ru - доменное имя, название узла (хоста) и дополнительные параметры.
* /direktoriya/ - директория, в которой располагается нужный нам файл. Из этой части ссылки мы видим, что на сервере хранится папка direktoriya, которая содержит необходимый нам файл.
* fail.html - файл с расширением ".html", он-то нам и нужен.

Как видно, папки и файлы разграничиваются знаком "/", который называется "слеш". Путь может быть длиннее. Кроме этого доменное имя может быть указанно с WWW и без WWW. Т. е. может быть: http://www.webmastermix.ru. Это не обязательный параметр, поэтому без разницы укажете вы его или нет необходимый вам файл все равно откроется. Для упрощения его лучше не использовать. И если ваш сайт доступен по двум адресам т. е. с WWW и без WWW, то лучше склеить эти URL и привести к одному виду, где WWW не будет использоваться. Как это сделать читайте в статье: [Как настроить 301 редирект в htaccess и в скриптах - более 18 примеров использования](http://webmastermix.ru/seo-optimization/261-kak-nastroit-301-redirekt-v-htaccess.html).

В общем и целом, что такое url сайта, страницы или изображения ясно. А теперь немножко углубимся.

**Структура URL-адреса**

**Общая схема (структура) URL-адреса следующая:**



**Каждый из этих параметров важен и имеет своё значение:**

* Схема – протокол передачи данных. Через него мы обращаемся к ресурсу. (http, https, ftp и т.д.)
* Логин и пароль – эти данные в протоколе http, как правило, не указываются. Они определяют права пользователя на доступу к определенной странице сайта.
* Собака (@) – специальный разделитель. Без логина и пароля его не указывают.
* Хост – это доменное имя, иногда IP-адрес ресурса, к которому обращается пользователь. (webmastermix.ru)
* Порт тоже указывается довольно редко. При обращении к сайту или любому файлу размещенному на сайте автоматически присваивается одно из значений – 80 или 8080.
* URL-путь – путь к тому или иному файлу. (/direktoriya/fail.html)
* Параметры – запрос с определёнными параметрами, которые передаются на сервер методом GET. Чтобы разделять параметры применяют знак &, например: ?параметр1=значение1&параметр2=значение2
* Якорь – это уникальная строка, состоящая из букв и/или цифр. Она ссылается на определённую область открываемого веб-документа. С его помощью можно сделать так, чтобы пользователь попал не в начало HTML документа, а сразу к определенному месту.

**Дополнительная информация касающаяся URL адресов**

**1.** В URL адресах используются различные кодировки. Если перекодирования нет, использовать можно только определённое количество символов. Рекомендуется использовать символы [0-9],[a-z],[A-Z],[\_],[-].

Чтобы избежать ошибок, разработчик должен именовать файлы сайта только в нижнем регистре и путь к ним указывать точно так же – маленькими буквами. Потому как, на юникс-подобных системах, а на них, чаще всего, работают веб-серверы, символы, написанные в разных регистрах, будут восприниматься машиной как разные. В Windows такого нет, но пренебрегать правилом, всё же, не стоит потому как если ваш файл назван в нижнем регистре, а в URL его название вы указали в верхнем регистре, то сервер не сможет его открыть.

Русские символы использовать можно, однако каждый русский символ будет проходить перекодировку (URL Encoding). И этой после перекодировки они будут выглядеть «страшно», ведь любой символ кириллицы будет закодирован с помощью 2 байт в UTF-8 в шестнадцатеричном виде. Разделяются символы знаком «%».

**Например ссылка вида:**

http://webmastermix.ru/всё для веб-мастера

**Будет иметь вид:**

http://webmastermix.ru/%u0432%u0441%u0435%20%u0434 %u043B%u044F%20%u0432%u0435%u0431%u043C%u0430 %u0441%u0442%u0435%u0440%u0430

Бороться с таким неудобством пытаются, однако, не слишком быстро. Поэтому, используя ЧПУ на своих CMS, лучше использовать транслитерацию (vse-dlya-webmastera). Ориентироваться будет проще, да и  для SEO все же наверное лучше.

**2.** Есть ещё такие URL - адреса которые называются динамическими. Такие URL содержат разные символы («?», «&», «=» и пр.) при помощи которых задаются параметры, кода мы рассматривали структуру URL уже упоминали про параметры, смотрите выше. Появляются такие адреса вследствие работы скриптов, размещённых на сайте. В таком случае, ссылка может иметь примерно такой вид:

http://webmastermix.ru/?page\_id=1

**3.** Кроме этого URL страницы может содержать якорь (о нем тоже шла речь выше, когда рассматривалась структура URL) и иметь следующий вид:

http://webmastermix.ru/direktiriya/fail.html#1razdel

Как видите здесь на конце ссылки появился еще один параметр - #1razdel, который называется якорем ссылки. С этим вы так же часто будете сталкиваться при создании сайта и материалов. При помощи URL в котором указан якорь мы можем автоматически переместиться к той области HTML страницы где данный якорь указан. Очень часто такие URL используют на страницах с большим количеством контента, для упрощения навигации по такой странице создается содержание состоящее из таких ссылок.

Чтобы создать такую ссылку, нужно заранее, в то место HTML страницы, куда вы хотите перенаправлять по ней пользователя, вставить следующий HTML код:

<a name="1razdel"> </a>

После этого можно сослаться на эту страницу при помощи ссылки представленной выше.

2) Основные понятия. Взаимодействие браузера и сервера

Когда вводится какой-нибудь **URL** в строку адреса браузера, браузер извлекает из **URL** протокол (например, **http**) и имя сервера. Согласно полученной информации он обращается к указанному серверу. Протокол — это набор правил, согласно которым браузер будет общаться с **Web**-сервером (если используется протокол **http**).

Компьютер — это «железо», которое ничего не может сделать без программного обеспечения. На компьютере, к которому обращается браузер должна быть установлена программа -- **Web**-сервер, которая в действительности будет обслуживать запросы. Когда мы говорим сервер, должны иметь в виду программу, а не компьютер.

Браузер определил, что имя сервера есть в базе данных **DNS**, узнал его **IP**-адрес и, используя его, подключился к серверу. Сервер ждет команды от браузера. Браузер посылает запрос — что-то наподобие: «Мне нужен документ **/path/docI.html**». Сервер просматривает содержимое каталога **/path**, находит там файл **docl .html** и возвращает его браузеру, как обыкновенный текст. Кроме текста сервер также возвращает тип документа. Бразуер отображает документы согласно информации о типе, переданной сервером. Например, если сервер скажет, что файл **docl.html** является текстом, то в окне браузера мы увидим код **HTML**, но обычно сервер так не делает, и мы видим нормальную **Web**-страницу — с картинками и форматированием.

Если запрашиваемый документ не найден, сервер может возвратить документ, содержащий сообщение об ошибке или просто код ошибки, например, 500 — внутренняя ошибка сервера. В последнем случае, браузер, получив код ошибки, возвращает встроенное в него сообщение об этой ошибке.

3) Основные понятия. Протокол HTTP

|  |
| --- |
|  |
|  |  |  |
|  |

**HTTP** — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).  
  
Аббревиатура HTTP расшифровывается как *HyperText Transfer Protocol*, «протокол передачи гипертекста». В соответствии со спецификацией [OSI](http://en.wikipedia.org/wiki/OSI_model), HTTP является протоколом прикладного (верхнего, 7-го) уровня. Актуальная на данный момент версия протокола, HTTP 1.1, описана в спецификации [RFC 2616](http://tools.ietf.org/html/rfc2616).  
  
Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом.  
  
Задача, которая традиционно решается с помощью протокола HTTP — обмен данными между пользовательским приложением, осуществляющим доступ к веб-ресурсам (обычно это веб-браузер) и веб-сервером. На данный момент именно благодаря протоколу HTTP обеспечивается работа Всемирной паутины.  
  
Также HTTP часто используется как протокол передачи информации для других протоколов прикладного уровня, таких как SOAP, XML-RPC и WebDAV. В таком случае говорят, что протокол HTTP используется как «транспорт».  
  
API многих программных продуктов также подразумевает использование HTTP для передачи данных — сами данные при этом могут иметь любой формат, например, XML или JSON.  
  
Как правило, передача данных по протоколу HTTP осуществляется через TCP/IP-соединения. Серверное программное обеспечение при этом обычно использует TCP-порт 80 (и, если порт не указан явно, то обычно клиентское программное обеспечение по умолчанию использует именно 80-й порт для открываемых HTTP-соединений), хотя может использовать и любой другой.

#### Как отправить HTTP-запрос?

Самый простой способ разобраться с протоколом HTTP — это попробовать обратиться к какому-нибудь веб-ресурсу вручную. Представьте, что вы браузер, и у вас есть пользователь, который очень хочет прочитать статьи Анатолия Ализара.  
  
Предположим, что он ввёл в адресной строке следующее:  
  
http://alizar.habrahabr.ru/  
  
Соответственно вам, как веб-браузеру, теперь необходимо подключиться к веб-серверу по адресу alizar.habrahabr.ru.  
  
Для этого вы можете воспользоваться любой подходящей утилитой командной строки. Например, telnet:  
  
telnet alizar.habrahabr.ru 80  
  
Сразу уточню, что если вы вдруг передумаете, то нажмите Ctrl + «]», и затем ввод — это позволит вам закрыть HTTP-соединение. Помимо telnet можете попробовать nc (или ncat) — по вкусу.  
  
После того, как вы подключитесь к серверу, нужно отправить HTTP-запрос. Это, кстати, очень легко — HTTP-запросы могут состоять всего из двух строчек.  
  
Для того, чтобы сформировать HTTP-запрос, необходимо составить стартовую строку, а также задать по крайней мере один заголовок — это заголовок Host, который является обязательным, и должен присутствовать в каждом запросе. Дело в том, что преобразование доменного имени в IP-адрес осуществляется на стороне клиента, и, соответственно, когда вы открываете TCP-соединение, то удалённый сервер не обладает никакой информацией о том, какой именно адрес использовался для соединения: это мог быть, например, адрес alizar.habrahabr.ru, habrahabr.ru или m.habrahabr.ru — и во всех этих случаях ответ может отличаться. Однако фактически сетевое соединение во всех случаях открывается с узлом 212.24.43.44, и даже если первоначально при открытии соединения был задан не этот IP-адрес, а какое-либо доменное имя, то сервер об этом никак не информируется — и именно поэтому этот адрес необходимо передать в заголовке Host.  
  
Стартовая (начальная) строка запроса для HTTP 1.1 составляется по следующей схеме:  
  
[Метод](https://habr.com/ru/post/215117/#method) [URI](https://habr.com/ru/post/215117/#uri) HTTP/[Версия](https://habr.com/ru/post/215117/#requestversion)  
  
Например (такая стартовая строка может указывать на то, что запрашивается главная страница сайта):  
  
GET / HTTP/1.1  
  
**Метод** (в англоязычной тематической литературе используется слово *method*, а также иногда слово *verb* — «глагол») представляет собой последовательность из любых символов, кроме управляющих и разделителей, и определяет операцию, которую нужно осуществить с указанным ресурсом. Спецификация HTTP 1.1 не ограничивает количество разных методов, которые могут быть использованы, однако в целях соответствия общим стандартам и сохранения совместимости с максимально широким спектром программного обеспечения как правило используются лишь некоторые, наиболее стандартные методы, смысл которых однозначно раскрыт в спецификации протокола.  
  
**URI** (*Uniform Resource Identifier*, унифицированный идентификатор ресурса) — путь до конкретного ресурса (например, документа), над которым необходимо осуществить операцию (например, в случае использования метода GET подразумевается получение ресурса). Некоторые запросы могут не относиться к какому-либо ресурсу, в этом случае вместо URI в стартовую строку может быть добавлена звёздочка (астериск, символ «\*»). Например, это может быть запрос, который относится к самому веб-серверу, а не какому-либо конкретному ресурсу. В этом случае стартовая строка может выглядеть так:  
  
OPTIONS \* HTTP/1.1  
  
**Версия** определяет, в соответствии с какой версией стандарта HTTP составлен запрос. Указывается как два числа, разделённых точкой (например **1.1**).  
  
Для того, чтобы обратиться к веб-странице по определённому адресу (в данном случае путь к ресурсу — это «/»), нам следует отправить следующий запрос:  
  
GET / HTTP/1.1  
Host: alizar.habrahabr.ru  
  
При этом учитывайте, что для переноса строки следует использовать символ возврата каретки (Carriage Return), за которым следует символ перевода строки (Line Feed). После объявления последнего заголовка последовательность символов для переноса строки добавляется дважды.  
  
Впрочем, в спецификации HTTP рекомендуется программировать HTTP-сервер таким образом, чтобы при обработке запросов в качестве межстрочного разделителя воспринимался символ LF, а предшествующий символ CR, при наличии такового, игнорировался. Соответственно, на практике бо́льшая часть серверов корректно обработает и такой запрос, где заголовки отделены символом LF, и он же дважды добавлен после объявления последнего заголовка.  
  
Если вы хотите отправить запрос в точном соответствии со спецификацией, можете воспользоваться управляющими последовательностями \r и \n:  
  
echo -en "GET / HTTP/1.1\r\nHost: alizar.habrahabr.ru\r\n\r\n" | ncat alizar.habrahabr.ru 80

#### Как прочитать ответ?

Стартовая строка ответа имеет следующую структуру:  
  
HTTP/[Версия](https://habr.com/ru/post/215117/#responseversion) [Код состояния](https://habr.com/ru/post/215117/#statuscode) [Пояснение](https://habr.com/ru/post/215117/#reasonphrase)  
  
**Версия** протокола здесь задаётся так же, как в запросе.  
  
**Код состояния** (*Status Code*) — три цифры (первая из которых указывает на класс состояния), которые определяют результат совершения запроса. Например, в случае, если был использован метод GET, и сервер предоставляет ресурс с указанным идентификатором, то такое состояние задаётся с помощью кода 200. Если сервер сообщает о том, что такого ресурса не существует — 404. Если сервер сообщает о том, что не может предоставить доступ к данному ресурсу по причине отсутствия необходимых привилегий у клиента, то используется код 403. Спецификация HTTP 1.1 определяет 40 различных кодов HTTP, а также допускается расширение протокола и использование дополнительных кодов состояний.  
  
**Пояснение** к коду состояния (*Reason Phrase*) — текстовое (но не включающее символы *CR* и *LF*) пояснение к коду ответа, предназначено для упрощения чтения ответа человеком. Пояснение может не учитываться клиентским программным обеспечением, а также может отличаться от стандартного в некоторых реализациях серверного ПО.

#### А что с безопасностью?

Сам по себе протокол HTTP не предполагает использование шифрования для передачи информации. Тем не менее, для HTTP есть распространённое расширение, которое реализует упаковку передаваемых данных в криптографический протокол **SSL** или **TLS**.  
  
Название этого расширения — **HTTPS** (*HyperText Transfer Protocol Secure*). Для HTTPS-соединений обычно используется TCP-порт 443. HTTPS широко используется для защиты информации от перехвата, а также, как правило, обеспечивает защиту от атак вида **man-in-the-middle** — в том случае, если сертификат проверяется на клиенте, и при этом приватный ключ сертификата не был скомпрометирован, пользователь не подтверждал использование неподписанного сертификата, и на компьютере пользователя не были внедрены сертификаты центра сертификации злоумышленника.  
  
На данный момент HTTPS поддерживается всеми популярными веб-браузерами.

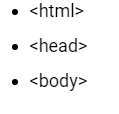
4) HTML. Базовая структура документа. Теги блока head.

Структура всегда одна и та же. Если вы захотите поменять что-то, браузер не сможет это обработать. В результате вы не получите то, что задумали.

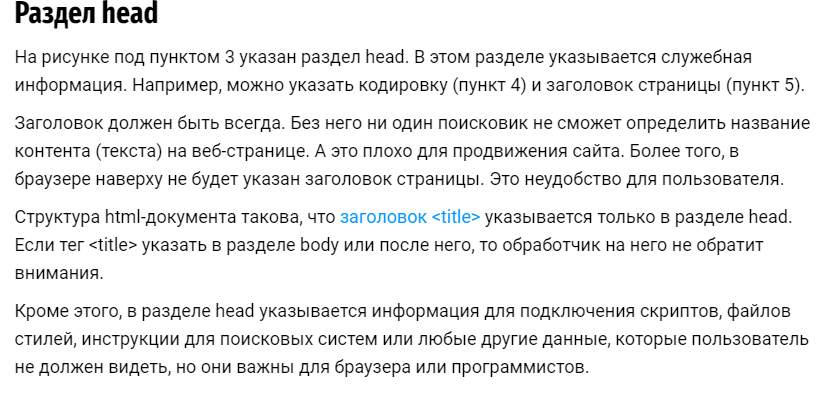


На рисунке выше указана структура любого html-файла. Первый пункт указывает на тип файла. Этот тег указывается один раз. Если вы будете использовать специальные редакторы, то вся структура создастся автоматически. Вам нужно будет подправить стандартные значения.

Структура html-документа - основные теги:



Из этих трех тегов состоит каркас всего сайта. Обратите внимание на рисунок. Все эти теги имеют закрывающийся тег со знаком “/”. Если вы пишете от руки, привыкайте ставить сразу оба тега – открывающий и закрывающий. Выше было сказано, что страницы сайтов имеют расширение .html. То есть если вы создадите текстовый документ, но при этом напишете правильный код, браузер всё равно отобразит вам просто текст. Никакого преобразования кода не будет.

****