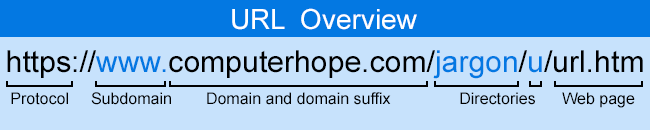
URL Rewriting

# Какво е URL?

[1] <https://www.computerhope.com/jargon/u/url.htm>

[2] <https://www.huify.com/blog/what-is-a-url-and-how-does-it-work>

URL (Uniform Resource Locator), също познат като интернет адрес, е вид URI и стандартизирана конвенция за именуване на документи, които са достъпни в Интернет.   
URL има два основни компонента – идентификатор на протокола и име на ресурса, които са разделени от двуеточие ( : ) и две наклонени черти ( // ).



На картинката е представено от какво е изграден един URL.

„http” отговаря за Hypertext Transfer Protocol. Той показва на браузъра кой протокол да ползва, за да достъпи определена информация в домейна. Когато обаче е https, това означава, че информацията, която се праща чрез HTTP, е криптирана и сигурна – т.е secure, откъдето идва и допълнителното s. Други примери за протоколи са File Transfer Protocol (FTP), Gopher и т.н Въпреки наличието на дрги протоколи, HTTPS е най-често срещаният и се е превърнал в стандарт за днешните уеб сайтове.

Следващата част от URL е “www“ означаваща World Wide Web или така нареченият поддомейн (subdomain). Използва се за логическо разделение на даден уебсайт на категории. „www“ не е задължителна и може да се пропусне. Т.е дали ще напишете [http://computerhope.com](https://computerhope.com/) или <http://www.computerhope.com>, крайният резултат ще е един и същ.

Следваща поред част е домейнът (root domain) и наставката (domain suffix). Домейнът е основният идентификатор на сайта. За да можем да намерим даден сайт, е много важно правилно да сме изписали домейна. Наставката, или още Top-Level-Domain (TLD), идентифицира характеристика на сайта – неговата цел, организацията, която го притежава, или географската област, откъдето произлиза. Например “.com” произлиза от commercial (търговски), „.org“ от organization (организация), а “.co.uk” показва местоположение, а именно United Kingdom (Великобритания).

“/jargon/u/” в примера от картинката са директории (subfolder), показващи къде се намира тази страница в сървъра. Това са заявки към страницата да предостви определена категория или тип страница.

На последно място в този пример е „url.htm”, което показва реалната страница от домейна, която сме посетили. „.htm“ отговаря на разширението на файла, в случая HTML файл. Други срещани разширения са .html, .php, .xml, .gif и т.н. Както различните типове файлове на компютъра ни, така и тези различни разширения изпълняват различна роля.

# Какво е URL Rewriting?

[3] <https://aloneonahill.com/blog/url-rewriting-for-beginners/>

[4] <https://en.wikipedia.org/wiki/Rewrite_engine>

[5] <https://www.smashingmagazine.com/2011/11/introduction-to-url-rewriting/>

Пренаписването на URL е един от най-добрите и бързи методи за подобряване на използваемостта на даден сайт, както и по-лесното му намиране. Повечето динамични сайтове съдържат променливи в своите адреси, с което казват на сайта каква информация да покаже на потребителя.

Чрез пренаписване на URL, самият адрес и ресурсите, до които води, могат да бъдат напълно независими един от друг. Макар и на практика да не са изцяло независими – URL адресът съдържа код, номер или име, което помага на системата за управление на съдържанието да потърси ресурса, на теория пренаписването предоставя пълно разделение.

Като всяко нещо, пренаписването на URL също има плюсове и минуси. Плюсовете включват по-изчистени и описателни адреси, улесняващи както потребителите, така и търсачката. Също така сайтът може да продължи да ползва същите URL адреси, дори и да смени технологията, която ползва. Като недостатък може да се приеме, че ако потребител се опита да достъпи нова информация чрез модифициране на дадения URL, пренаписването може да възпрепядства конструкцията на персонализирани заявки заради липса на именувани променливи.

# Как се пренаписва URL?

[5] <https://www.smashingmagazine.com/2011/11/introduction-to-url-rewriting/>

[3] <https://aloneonahill.com/blog/url-rewriting-for-beginners/>

Правилата за пренаписване формират част от конфигурацията на уебсайт на Apache сървъра и могат да бъдат поставени на различни места като част от виртуалната хост конфигурация.

Възможността да имплементираш пренаписване на URL зависи от Уеб сървъра. Например Apache идва с вече инсталиран и активиран модул за пренаписване – mod\_rewrite. От друга страна IIS, сървърът на Microsoft, не включва поодбен модул, но позволява добавянето на такава функционалност чрез добавки.

Ще ви представя първо един лесен пример, с който да минем през основите на правилата за пренаписване на URL чрез mod\_rewrite.

Като за начало, нека си представим следния пример – имаме уебсайт и един PHP скрипт за една страница. Нейният URL адрес е <http://www.pets.com/pet_care_info_07_07_2008.php>

Нашата цел е да изчистим този адрес и да го направим удобен за търсене от търсачката и помнене от потребителите. Идеалният URL в този случай би бил <http://www.pets.com/pet-care/>

За да работи това, трябва да кажем на сървъра вътрешно да пренасочи всички заявки за „pet-care“ към "pet\_care\_info\_07\_07\_2008.php". Искаме това да се случи вътрешно, защото не искаме самият URL в полето на браузъра да се изменя.

За целта първо трябва да създадем текство документ с име „.htaccess“, който да съдържа нашите правила. Този документ ще бъде поставен в root директорията на сървъра, като ако вече има такъв документ е по-добре той да бъде редактиран, а не пренаписван.

Apache сървърът предоставя конфигураця чрез „.htaccess“ файлове. Тези файлове позволяват фино настройване на универсалите системни конфигурации на Apache, които са дефинирани в главния конфигурационен файл на Apache. Локализираните директиви трябва да оперират във файл на име „.htaccess“ като е важно потребителят да има необходимите разрешения за достъп и редактиране на този файл.

Отделно този файл не трябва да доставя world write access, т.е най-сигурната настройка за разрешение е „644“, която позволява универсално четене, но писане само от потребител. Правилата, написани в .htaccess се прилагат за parent директорията и всички поддиректории. За да се приложат правилата за целия сайт, файлът трябва да се постави в root директорията на сайта.

Файлът „.htaccess“ е конфигурационен за сървъра.Като такъв, той е много мощен и всяка грешка, дори и синтактична, може да доведе до неизправност. Ако има грешки, сървърът ще сигнализира – най-често с грешка с код „500“). Затова е важно да се правят резервни копия на всичко, свързано със сайта, включително и на оргиниалните htaccess файлове, преди някакви промени да бъдат направени. Използваме този файл с две цели – едната е да каже на Apache да включи пренаписващата машина (rewrite engine) и втората е да каже на Apache какво правило за пренаписване искаме да ползва. Във файла трябва да добавим следното:

RewriteEngine On # Turn on the rewriting engine

RewriteRule ^pet-care/?$ pet\_care\_info\_01\_02\_2008.php [NC,L] # Handle requests for "pet-care"

Забележка: Коментарите са важни за контролиране на всякакъв код. В този файл закоментирането става с помощта на #, като този знак се поставя преди всеки ред, който искаме да коментираме, т.е не може с # да се коментират множество редове. Също така RewriteEngine рябва да се ползва само веднъж в „.htaccess“ файл.

Същинското пренаписване се случва благодарение на реда RewriteRule. Можем да го разделим на 5 части:

RewriteRule – казва на Apache, че този ред се отнася само за едно правило

^/pet-care/?$ - това е шаблонът. Сървърът проверява адреса на всяка заявка да провери дали шаблонът съвпада. Ако да, Apache ще го размени със заместващата секция, която следва

pet\_care\_info\_01\_02\_2003.php – споменатата заместваща секция. При съвпадение Apache ползва този URL вместо заявения

[NC,L] – „флагове“, които казват на Apache как да приложи правилото, като в този случай NC отговаря за case-insensitive, т.е да не се влияе от малки и големи букви, а L казва да не се прилагат други правила, ако това е използвано

# Handle requests for "pet-care" – както споменахме, всичко след # е коментар и като всеки друг коментар не е задължителен, а само препоръчителен

# чрез регулярни изрази

[3] <https://aloneonahill.com/blog/url-rewriting-for-beginners/>

Горното правило позволява пренасочване на заявка за един URL, но по-интересно и полезно е когато се стигне до идентифициране и пренаписване на групи от URL, базирано на определени шаблони, които съдържат.

За онагледяване на казаното, нека ползваме отново горния пример. Да кажем, че искаме да пренапишем всички налични URL по начина, по който пренаписахме горния. Съществуващият адрес е <http://www.pets.com/show_a_product.php?product_id=7> и искаме да го представим по следния начин <http://www.pets.com/products/7/> . Очевидно вместо да пишем правило за всеки адрес, бихме предпочели да имаме едно правило, което да пренаписва всички адреси за идентификационния номер на продуктите. Т.е искаме от http://www.pets.com/show\_a\_product.php?product\_id={a number} да стане http://www.pets.com/products/{a number}/.

Това е възможно чрез така наречените регулярни изрази. Това са шаблони, дефинирани в определен формат, който сървърът да може да разбере и да обработи правилно. Стандартен шаблон за избиране на числа изглежда така: „[0-9]+“, като в квадратните скоби се съдържа някакъв интервал от знаци, а 0-9 показва всички цифри. Плюсът ( + ) от своя страна показва един или много предхождащи знаци, т.е този шаблон означава „една или повече цифри“.

Улеснение от страна на сървъра е, че целият шаблон се третира като регулярен израз по подразбиране и няма нужда да се включва или активира допълнително.

Цялостният вид на правилото би бил:

RewriteRule ^products/([0-9]+)/?$ show\_a\_product.php?product\_id=$1 [NC,L] # Handle product requests

Важно уточнение при писането на регулярни изрази е наличието на така наречени „специални“ знаци. Това са знаци, които имат различно от обичайното значение за регулярния израз.

Някои от тези специални знаци включват:

. (any character)

\* (zero of more of the preceding)

+ (one or more of the preceding)

{} (minimum to maximum quantifier)

? (ungreedy modifier)

! (at start of string means "negative pattern")

^ (start of string, or "negative" if at the start of a range)

$ (end of string)

[] (match any of contents)

- (range if used between square brackets)

() (group, backreferenced group)

| (alternative, or)

\ (the escape character itself)

# flags

Флаговете се добавят в края на правилото, за да кажат на Apache как да интерпретира и управлява правилото. Поставят се в квадратни скоби [ ] и се разделят със запетая ( , ).

Почти всеки флаг има кратка и дълга форма. Въпреки че най-често се използва кратката форма е добре да се знаят пълните форми, защото те улесняват запомнянето на предназначението на всеки флаг. Някои флагове приемат един или повече аргументи и не се влияят от малки и големи букви.  
Ето и пълен списък с флаговете и техните значения:

C –

CO=cookie –

E=var:value –

F –

G –

H=handler –

L –

N –

NC –

NE –

NS –

P –

PT –

R –

R=301 –

QSA –

S=x –

T=mime-type –

# условия

Правилата за пренаписване могат да бъдат предшествани от едно или повече условия и те могат да се пишат заедно. Това позволява да се приложат определени правила за определни заявки.

Ето и пример за по-добро представяне на условията и как работят:

RewriteCond %{HTTP\_HOST} ^addedbytes.com [NC]

RewriteRule ^(.\*)$ http://www.addedbytes.com/$1 [L,R=301]

Горното правило пренасочва всички заявки, без значение за какво, към същия URL на [www.addedbytes.com](http://www.addedbytes.com). Без условието, правилото ще създаде цикъл като всяка заявка, която съвпада с правилото ще се праща към себе си. Така написано правилото е предназначено да пренасочва само заявки, където липсва “www” частта и условието подсигурява това да се случи.

Условието работи по сходен начин като правилото. Започва с RewriteCond, което казва на mod\_rewrite, че този ред се отнася за условие. Следва какво реално ще бъде тествано и шаблонът, по който ще се тества. На последно място се поставя флагът, точно като при RewriteRule.

Стрингът, който се тества, т.е втората част от условието, може да бъде множество различни неща. В горния пример се тества домейнът, към който се праща заявката. Може също да се тества браузъра, IP адресът на потребителя и т.н.

Шаблонът е почти същият като при правилото. Той може да не се интерпретира като такъв ако започва с определени знаци, например <,>,-.

Към условията могат да се добавят флагове, както при правилата, но те са само два – NC и OR. Вторият се използва когато искаме да приложим правило ако едно от две условия съвпада и вместо да повтаряме условието, добавяме флаг OR. Стандартното поведение на правилото, когато е предшествано от много условия, е, че ще се изпълни, ако всички условия съвпадат.

# Изключения и специални случаи

Условията за пренаписване на URL могат да бъдат тествани по няколко различни начина – няма нужда да бъдат третирани като регулярни изрази, въпреки че предимно така се ползват.

Ето и няколко начина за обработване на условия:

<Pattern

>Pattern

=Pattern

-d

-f

-s

-l

-F

-U

# Няколко прости примера

Като за начало нека разгледаме няколко прости примера за правила, изпълняващи стандартни команди

За пренасочване на стар домейн към нов:

RewriteCond %{HTTP\_HOST} old\_domain.com [NC]

RewriteRule ^(.\*)$ http://www.new\_domain.com/$1 [L,R=301]

За пренасочване на всички заявки, където липсва “www”

RewriteCond %{HTTP\_HOST} ^domain.com [NC]

RewriteRule ^(.\*)$ http://www.domain.com/$1 [L,R=301]

За пренасочване на всички заявки, съдържащи „www“

RewriteCond %{HTTP\_HOST} ^www.domain.com [NC]

RewriteRule ^(.\*)$ http://domain.com/$1 [L,R=301]

За пренасочване на стара страница към нова:

RewriteRule ^old-url.htm$ http://www.domain.com/new-url.htm [NC,R=301,L]

# Преместване на съдържание

Нека разгледаме следния пример:

RewriteRule ^article/?$ http://www.new-domain.com/article/ [R,NC,L] # Temporary Move

Чрез добавяне на флага R се изменя работата на правилото. Вместо да се пренаписва вътрешно, Apache праща съобщение обратно към браузъра (HTTP хедър), което казва дали документът е преместен временно. Този хедър съдържа код (302), който показва, че преместването е временно.

Ако искаме да преместим за постоянно даден URL, към R флага се добавя „=301“, с което Apache казва на браузъра да приема преместването за постоянно. За разлика от R, R=301 казва на браузъра да представи новия адрес в адрес бара.

Нагледно това правило изглежда така:

RewriteRule ^article/?$ http://www.new-domain.com/article/ [R=301,NC,L] # Permanent Move

Това е и най-често ползвания метод за пренаписване на URL на елементи, които са преместени на нови адреси.

# Сървърни променливи (Server Variables)

Сървърните промениви са селекция от елементи, които могат да бъдат тествани когато се пишат правила. Това позволява да се прилагат правила, базирани на всякакви заявени параметри, включително идентификатори на браузъра, препращащи адреси или множество от други стрингове.

Променливите се записват по следния начин:

%{VARIABLE\_NAME}

А VARIABLE\_NAME може да бъде заменено например с HTTP\_PROXY\_CONNECTION / SERVER\_ADMIN / TIME\_YEAR и т.н.

# Работа с много правила

Колкото по-сложни са сайтовете, толкова повече се усложняват и правилата. Това създава проблем, когато трябва да се решават конфликти между правила. Ако правилото само по себе си е правилно написано, то тогава има друго правило преди това, което съвпада с адреса и така адресът не се тества за новото добавено правило.

Ще използвам 2 примера за по-добро представяне на проблема:

Първият:

RewriteRule ^([A-Za-z0-9-]+)/([A-Za-z0-9-]+)/?$ get\_product\_by\_name.php?category\_name=$1&product\_name=$2 [NC,L] # Process product requests

RewriteRule ^([A-Za-z0-9-]+)/([A-Za-z0-9-]+)/?$ get\_blog\_post\_by\_title.php?category\_name=$1&post\_title=$2 [NC,L] # Process blog posts

В този пример продуктовите страници и блога имат идентични шаблони. Второто правило няма да съвпадне с URL, защото всеки адрес, който би съвпаднал вече е съвпаднал с горното правило.

Един начин за справяне с този проблем е като се добави допълнителна част към URL за обозначаване на типа заявка. Това се пише по следния начин:

RewriteRule ^products/([A-Za-z0-9-]+)/([A-Za-z0-9-]+)/?$ get\_product\_by\_name.php?category\_name=$1&product\_name=$2 [NC,L] # Process product requests

RewriteRule ^blog/([A-Za-z0-9-]+)/([A-Za-z0-9-]+)/?$ get\_blog\_post\_by\_title.php?category\_name=$1&post\_title=$2 [NC,L] # Process blog posts

Пример №2:

В някои случаи тази ситуация може да се избегне като се напишат по-точни правила.

Във втория пример са представени две правила, където се архивират постове по тема и по дата.

RewriteRule ^([A-Za-z0-9-]+)/?$ get\_archives\_by\_topic.php?topic\_name=$1 [NC,L] # Get archive by topic

RewriteRule ^([A-Za-z0-9-]+)/?$ get\_archives\_by\_year.php?year=$1 [NC,L] # Get archive by year

Така написано, правилата ще са в конфликт. За да се реши този проблем, може правилото за годините да се напише по-конкретно, все пак годините за 4-цифрени числа, и да се изпълни първо. Това би изглеждало така:

RewriteRule ^([0-9]{4})/?$ get\_archives\_by\_year.php?year=$1 [NC,L] # Get archive by year

RewriteRule ^([A-Za-z0-9-]+)/?$ get\_archives\_by\_topic.php?topic\_name=$1 [NC,L] # Get archive by topic