<IMAGE>

URL rewriting

Техника, наречена пренаписване на URL адреси, която може да превърне трудните за запомняне и четене URL адреси в добре разбираеми и лесни за възпроизвеждане , избягвайки разходи на време/финанси в измисленето на оптимален домейн. Позволява ви да попълните вашите URL адреси с удобни, четливи ключови думи, без да засягате основната структура на вашите страници.

## Въведение в URL [6][10]

Всеки един потребител на Интернет услуги и браузър поне веднъж чувал за термин URL и сигурно е използвал това “URL” за достъпване на HTML страници. Въпреки това, мнозинство не знае нищо за това съкращение, а час от тези, които знаят, само частично са запознати с URL.

Най-често хора си представят URL като името на конкретен файл на някоя машина в Интернет мрежа(WWW). Това не е изцяло правилно. Защо ? Защото URL може да сочи и до другите ресурси в мрежата като заявки към базата данни.

Пример:

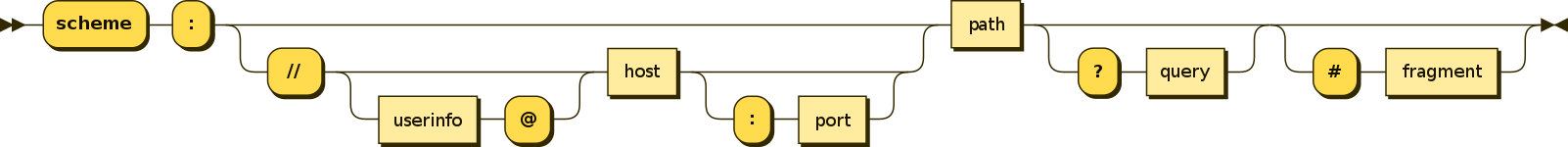
https://example.com/book?isbn=9780451524935

По горният пример показва заявката към базата данни, където искаме да получим книгата(book) със значението на променлива “isbn” равно на **9780451524935**.

Нека да дадем дефиниция за URL. URL(Uniform Resource Locator) – това е референция(адрес) към някакъв ресурс, намиращ се в мрежата Интернет. Или казано по прост начин – URL адресите са езикът за сърфиране в Интернет, което всеки ден осъществяваме. Самият URL може да се види в браузъра, а по-точно в адресната лента.

Хората могат да използват взаимозаменяемо понятията “URI” и вече споменато “URL”, но всъщност добре е да се прави разлика, че URL е специфичен вид на [URI](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier)(Uniform Resource Identifier) [17].

## Формат на URL адрес [10][11]



*Фигура-1*

**scheme:** схема или по-директно протокол, който браузър ще трябва да ползва да достъпи съответен ресурс. Самото името на протокол е непразна последователност на символи, започваща с буквата и следвана от някаква комбинация на букви/цифри/”+”/”.”/”-”. Името на схема е нечувствително към регистъра, но въпреки това каноничният начин за за изписването е точно с малки букви(http, https, mailto, ftp, data …).

**userinfo:** незадължителен подкомпонент на URL, който състои от потребителско име и парола, разделени със символ “:” (username:password).

**host:** подкомпонент, който е представен чрез името на домейн(www.example.com) или IP-адрес(IPV-4 или IPV-6).

**port:** портът е числова стойност, която се използва за идетифициране на техническа “порта” за достъп до ресурсите на уеб сървъра. Е незадължителен подкомпонент, ако се използват стандартните значения (за HTTP е 80, за HTTPS е 443). Иначе е задължително да се зададе явно.

**path:** път е последователност от имената на папките, разделени със символ “/”, която се завършва с името на “конкретен файл” с базовите расширения като: .html, .php, … Преди време подкомпонент path идентифицираше физическа локация на файл, намиращ се на уеб сървъра. Сега вече път от типа “/path/to/myfile/test.html” не е нищо повече от абстракция на уеб сървъра, която не сочи до физически файл.

**query:** заявка или параметри за един URL представляват от себе последователност от двойките ключ-стойност, които са разделени с разделител “&”. Параметрите за URL са незадължителни. Самата роля на тези параметри е такава, че уеб сървър може да извърши допълнителни действия преди да е върнал самият ресурс.

Пример:

https://api.example.com/products?category=electronics&page=1

По-горният пример показва, че искаме да достъпим ресурс, отнасящ се до products с категория electronics и страницата да е 1

**fragment(anchor):** този подкомпонент изпълнява роля на котва/отметка, премествайки потребителя точно до конкретната секция/част, която я е задал. За видеата това може да значи да прелистне видеото до конкретен момент на време.

Пример:

https://www.example.com/page#section6

Горният пример се опитва да достъпи ресурса page от уеб сървъра и е указан фрагмент “#section6”, към който потребителя иска да го навигират.

## За какви цели може да се ползва URL [10]

URL адресите се най-често служат като референция към уеб страници (HTTP/HTTPS), но се използват и за прехвърляне на файлове (FTP), имейли (mailto), достъп до база данни (JDBC) и много други приложения.

### URL и mailto[3]

Докато стандартните URL протоколи отварят новата страница във УЕБ, то mailto има за цел друго. Той отваря пощенски клиент по подразбиране, задавайки значения в съответните полета, които сме му подали като параметрите на заявка.

**Синтаксис:**

mailto:[email\_address]?[parameters]

**[email\_address]**: в тази секция се посочват имената на получатели на имейл, както ако са зададени няколко адреса, то се те се разделят със “,”.

**[parameters]:** в тази част на URL се задават двойки ключ-стойност за допълнителни параметри на един имейл като: CC, BCC, subject, body. Самите двойки се разделят със “&”.

Пример:

<a

       href="mailto:web@fakeemail.com, weba@fakeemail.com?cc=jackdoe@fakeemail.com

       &bcc=jennydoe@fakeemail.com

       &subject=this is how to use the mailto link

       &body=this is an article on how to use the mailto link">

       Example of mailto request

  </a>

### URL и JDBC[4]

JDBC или Java Database Connectivity е API(програмен интерфейс), който позволява Java приложения да комуникират с базите данни, изпращайки SQL-валидни заявки. JDBC дава възможност за извършване на стандартни [CRUD](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) [18] операции, но преди това трябва да се построи връзката с конкретната база данни. Това се извършва посредством изпращането на специфичен URL.

**Свързване на JDBC със MySQL**

Шаблон:

protocol//[hosts][/database][?properties]

Пример:

jdbc:mysql://mysql.db.server:3306/my\_database?useSSL=false&serverTimezone=UTC

“jdbc:mysql:”: е протокол, чрез който ще се изгради връзката.

“mysql.db.server”:е хост или по-точно казано адрес на самата базата данни, с която се опитаме да изградим конекция.

“3306”: конкретен порт, на който MySQL(в случая) работи.

“my\_database”: това е името на базата данни, с която се опитваме да се свържем.

“useSSL=false” и “serverTimezone=UTC”: това са параметрите на връзката, които се предават към базата данни.

## Има ли нужда от преработка на URL адреси [13][11]

URL адресите не винаги са представени в добре организиран и лесно четим от хора формат. Често се случва обратното. Самият URL може да съдържа името на скрипт(примерно **example.php**), който се разполага на сървъра. Този факт при промяна на реализация на скрипта може да наложи необходимост да се променя и самият URL. Също така, при подобните практики, то при преместване или реструктуризация на базата данни ще се появят счупени линкове(broken links) както вътрешни, така и от външни сайтове.

И се пита: “Има ли спасение от това ?”. Има ! Семантични URL адреси (Сlean URLs) са такива URL адреси, които са лесни за разпознаване от човек извън индустрията, оптимизирани са за търсачки (SEO, Search Engine Optimization) и допринасящи за REST стила за софтуерни архитектури. Казано по-просто: тези URL адреси отразяват същността на самият обект, който достъпват, вместо да ни ангажират с безрой ненужни параметри. Една от техниките за преработка на URL адреси е точно URL rewriting.

**Примери за най-чести преработки в URL адреси:**

|  |  |
| --- | --- |
| **Оригинален URL** | **“Чист” URL** |
| http://example.com/about.html | http://example.com/about |
| http://example.com/user.php?id=1 | http://example.com/user/1 |
| http://example.com/index.php?page=name | http://example.com/name |

Такава техни може и да оправи сякаш безнадеждно лошо формиран URL адрес като:

https://www.example.com/index.php?page=products&category=electronics&subcategory=laptops&product\_id=12345&ref=homepage&utm\_source=google&utm\_medium=cpc&utm\_campaign=spring\_sale&utm\_content=ad1&utm\_term=laptop\_deals&sessionid=abcdef1234567890.

И след URL rewriting адресът ще изглежда вече по следен начин:

https://www.example.com/products/electronics/laptops/12345

Впечатляващо. Доста дори. Затова самият механизм URL rewriting ще го разгледаме по-подробно.

## URL rewriting: концепция [5][1]

URL rewriting е техника, която модифицира изгледа на URL адреса с помощта на пренаписващ двигател(rewrite engine). Този двигател е най-често част от УЕБ сървър или WEB framework. Като резултат на тази техника URL адресите стават както по-разбираеми и по-четими за хора, така и по-лесни за обработка на търсачки(search engines).

Ако стандартната зависимост между адреса и ресурса е: **един адрес = един ресурс**, то URL rewriting добавя допълнителна абстракция до тази зависимост. Тази техника почти изцяло елиминира зависимост между ресурс(примерно УЕБ страница) и адреса. С нейното прилагане адресите почват да наподобяват символни линкове. Така зависимост **един адрес = един ресурс** става **един адрес = един начин да намериш ресурс**.

Следните адреси от примерите ще сочат до един и същи ресурс, което се осигурява през по-горе спомената абстракция.

Пример 1:

…/page1.html

…/about-us/

…/about-this-website-and-me/

…/guess-who-am-i-24242/

Пример 2:

http://www.example.com/Blog/Posts.php?Year=2006&Month=12&Day=19

http://www.example.com/Blog/2006/12/19/

## URL rewriting: Apache mod\_rewrite

Ще разгледаме URL rewriting механизма в контекста на модула на Apache сървъра, който се казва mod\_rewrite.

### С какво да започнем ? [15]

Преди да започнем, първо трябва да активираме на УЕБ сървъра Apache самият двигател за преработка(rewrite engine). Това ще стане чрез добавяне на конкретният ред до конфигурационен файл httpd.conf, който се намира в коренова директория на Apache.

AllowOverride All

След това httpd.conf трябва да изглежда подобно на:

A screenshot of a computer

Description automatically generated

*Фигура-2*

Слагайки “AllowOverride All” в конфигурационен файл на Apache, ние ще дадем възможност за презаписване на настройки(override settings). Това ще ни е необходимо, за да се извършва процес за пренаписване(пренасочване) на адреси.

Като следваща стъпка се създава файл .htaccess в коренова папка, в чиято йерархия ще седят УЕБ ресурси(html страници), чиито URL адреси ще преработваме.

Примерен път до коренова папка:

…/htdocs

Примерен път до УЕБ ресурс:

…/htdocs/web/referats/url/referat.html

В самият файл .htaccess добавяме ред за активация на модула Apache mod\_rewrite:

RewriteEngine on

И като пример ще достъпим един и същи referat.html файл през 2 различни URL адреса.

“.htaccess**”** файл:

RewriteEngine On

RewriteRule "**^referat$**" "**/web/referats/url/referat.html**" [L]

Достъпване на “referat.html” през абсолютен път:



*Фигура-3*

Достъпване на “referat.html” през съкратен израз:



*Фигура-4*

И двата линка водят до един и същи ресурс. Следователно mod\_rewrite заедно с rewrite engine изпълняват своите функции коректно.

**Забележка:** CSS стилове в по-горните файлове за пример са вградени чрез атрибути на тагове. Как да се извърши коректно пренасочване на URL адрес на файл, който е свързан с други външни файлов, това ще бъде показано в този пример.

### Как работи mod\_rewrite [9]

mod\_rewrite се базира на регуларните изрази([PCRE](https://en.wikipedia.org/wiki/Perl_Compatible_Regular_Expressions)). С помощта на изрази от специално значение(или точно зададен URL адрес) такива части на първоначален URL адрес като /path и ?query може да се пренапишат.

Ако във файл .htaccess няма явно зададен ред RewriteEngine on, то тогава преработващ двигател не се функционара. Вместо да коментираме целият код с инструкциите за преработка, ние си работим само с един ред, който определя състоянието на модула mod\_rewrite(дали ще работи).

### Базови регуларни изрази [9]

|  |  |
| --- | --- |
| **Символ** | **Значение** |
| . | Всякакъв единствен символ |
| + | Предишен символ/група се повтаря един или повече пъти |
| \* | Предишен символ/група се повтаря нула или повече пъти |
| ? | Предишен символ/група се повтаря нула или един път |
| \ | Превръща символи със специално значение в стандартни символи (escape character) |
| ^ | Определя началото на реда |
| $ | Определя края на реда |
| ( ) | Елементи, заградени в този символ, се групират и се счита, че ще са разположени точно в задана последователност |
| [ ] | Елементи/конструкции, заградени в този символ, се считат за клас и се счита, че един от всичките елементи от този клас ще трябва да се срещне в контекста на конкретна позиция |
| [^ ] | Всеки един елемент/конструкция може да се срещне в контекста на конкретна позиция, освен тези, които са специфирани вътре в конструкцията след символа “^” |
| { } | Определя брой повторения на елемент, който стои преди конструкцията “{ }”. Ако искаме предишният символ/група да се повтаря максимално N пъти, то {,N}. Да се повтаря минимум N пъти - {N}. Да се повтаря между N и M пъти – {N,M} |
| | | Има функция като логическото ИЛИ. Един от символите/конструкции трябва да се срещне на конкретна позиция (или символ отляво на “|” или отдясно) |

### Базов синтаксис [9][7]

.htaccess файл може да съдържа в себе си следните ключови думички: RewriteCond, RewriteRule и RewriteEngine.

Примерната структура на файла:

RewriteEngine On

RewriteCond %{HTPPS} off

RewriteRule "^referat$" "/web/referats/url/referat.html" [L]

RewriteCond клаузите могат да бъдат няколко подред и ако не е специфицирано друго, то следващото след тях RewriteRule ще се приложи, ако всичките условия пред него ще се изпълнят. Може да се уствнови релация между условията ИЛИ и тогава вече при изпълнението на поне едно условие ще бъде направена преработка на URL адрес.

RewriteEngine On

RewriteCond %{HTPPS} off **[OR]**

RewriteCond %{QUERY\_STRING} ^$

RewriteRule ^page\.html$ newpage.html

По-горният може да се тълкува по следен начин: ако https е изключен(http e протоколът) **ИЛИ** параметри на заявка(query string) не са зададени, то тогава търсим след path частта на URL page.html и го заменяме на newpage.html.

### RewriteCond [7]

С тази ключова дума задаваме, че искаме да има условие, при изпълнението на което да се прилага RewriteRule.

**Синтаксис:**

RewriteCond НизЗаТестване Условие [Флагчета]

НизЗаТестване – това е низ, който най-често отобразява сървърни променливи и те се получават със синтаксис %{VARIABLE}. Самият низ ще се тества в зависимост от оператора със следващият до него израз.

Условие – този низ определя начин, по който ще се тества предишният израз до него. Това може да бъде както регулиран израз, така и сравнението на двата низа: НизЗаТестване и Условие. Ако самият тест е преминал успешно, то се преминава успешно към преработката на URL адрес в частта RewriteRule.

[Флагчета] – този параметър не е задължителен. Се ползват с цел да модифицират зададено условие и изпълнението на RewriteCond изпълнението. Примерни ефекти са: [OR] между 2 RewriteCond означава, че поне един от тях трябва да е вярен, а не 2 задължително; [NC] при RewriteCond означава, че условието ще бъде тествано, игнорирайки регистъра (малки/големи букви ще се третират по един и същи начин).

### RewriteRule [7]

Ключовата дума RewriteRule определя начин за възможната преработка на URL адрес - свързване на низ със абсолютен файл от файловата система или конкретен URL.

**Синтаксис:**

RewriteRule Патерн Заместване [Флагчета]

Патерн – регуларен израз, който ще се прилага върху от частта path на URL адрес и до края на адреса.

Заместване – варианти за стойност на тази част са 3. При задаването на -, то никакви действия не се извършат по отношение към частта Патерн. При задаването на абсолютен път към конкретен файл от файловата система, то точно той ще бъде отворен при процедирането с URL с примерен синтаксис “http://hostname/Патерн”. Също така може да се зададе абсолютно нов URL адрес, към който да се извършва пренасочването.

[Флагчета] – този параметър не е задължителен. Се ползват с цел да модифицират зададено условие и изпълнението на пренаписване на URL адреси. При зададен флаг [L] задаваме, че ако текущият израз RewriteRule се изпълни успешно, то следващите изрази не се обработват. [NC] при RewriteRule означава, че Патерн ще бъде обработен, игнорирайки регистъра (малки/големи букви ще се третират по един и същи начин). При зададен флаг [R] казваме на сървъра, че искаме да заместим временно([R=302]) или постоянно([R=301]) истинският URL адрес. Ако заменяме адрес завинаги, то буквално индексираме наново адрес(презаписваме оригинален) и сървър ще знае, че не сме намерени да ползваме отново оригинален адрес. При временно пренасочване вече е друго – там сървър на някакъв срок ще пренасочи адреса, но пък възможност ще остане да се върнем към първоначалния URL адрес(обикновенно се ползва, когато искаме да изтестваме нещо).

### Regex back-reference [9]

При наличието на групите в регуларните изрази, които може да са част от RewriteCond или RewriteRule, може да преизползваме конкретните групи, които са били успешно намерени от регуларен израз.

**Примерен сценарий за използане:**

RewriteCond %{QUERY\_STRING} ^id=([0-9]+)$

RewriteCond %{REQUEST\_URI} ^/category/([^/]+)/?$

RewriteRule “^/?([a-zA-Z]+)/([a-zA-Z]+)/?$” “/newpage.php?category=%1&id=%2&group=$1&subgroup=$2 [R=301,L]”

Референции от вида **%N:** реферира N-та група от RewriteCond изрази или по-точното от тяхната част за Условие. Номерирането на групи се почва отзад напред(последно употребена група ще може да бъде достъпена с **%1**). **%0** връща низ със всичките съществуващи групи от съответните изрази.

Референции от вида **$N:** реферира N-та група от RewriteRule израза, в чиято част Заместване е специфицирана(тоест е ограничено до тялото на конкретен израз). Последно употребена група ще може да бъде достъпена с **%1**. **%0** връща низ със всичките съществуващи групи на съответен израз.

## mod\_rewrite: специфични примери

### Пример-1: няколко RewriteCond израза:

С променливата %{REQUEST\_URI} получаваме path частта на URL, а с променливата %{HTTP\_HOST} host частта на URL. Ако и двата условия са в сила, то тогава ще се продължи до RewriteRule.

RewriteEngine On

RewriteCond %{REQUEST\_URI} ^(\/?referat)$

RewriteCond %{HTTP\_HOST} ^(localhost)$

RewriteRule ^(\/?referat)$ /web/referats/url/$1.html

RewriteRule "^(referat\.css)$" "/web/referats/url/$1" [L]

### Пример-2: няколко RewriteCond и флаг [OR]:

С променливата %{REQUEST\_URI} получаваме path частта на URL, а с променливата %{HTTP\_HOST} host частта на URL. Ако поне едно условие е в сила, то тогава ще се продължи до RewriteRule. Това става заради флагчето [OR].

RewriteEngine On

RewriteCond %{REQUEST\_URI} ^(\/?referat)$ [OR]

RewriteCond %{HTTP\_HOST} !^(localhost)$

RewriteRule ^(\/?referat)$ /web/referats/url/$1.html

RewriteRule "^(referat\.css)$" "/web/referats/url/$1" [L]

### Пример-3: RewriteRule и [NC] флаг:

[NC] флаг добавя игнорирането на регистъра за буквите, което веднага прави всякакви възможни варианти за изписване на дума “referat”: rEFerat, REFerAt, … Такива пермутации може да са 2^7. И в резултата на всяка една пермутация крайният URL адрес ще бъде винаги /web/referats/url/referat.html, защото ползваме флаг [R=302] за временно пренасочване.

RewriteEngine On

RewriteCond %{REQUEST\_URI} ^(\/?referat)$

RewriteCond %{HTTP\_HOST} ^(localhost)$

RewriteRule ^(\/?referat)$ /web/referats/url/referat.html [NC,R=302,L]

### Пример-4: RewriteRule и релативно връзвани външни файлове:

Предположим, че следният файл “referat.html” включва външен CSS файл “referat.css” в себе си по следен начин:

<link *rel*="stylesheet" *href*="./referat.css">

Предположим, че се опитаме да пренасочим точно URL, който е свързан с “referat.html” файл.

RewriteEngine On

RewriteRule "^(referat)$" "/web/referats/url/$1.html"

На първи поглед, няма никакви проблеми. Обаче CSS файл, който беше свързан, вече след пренасочване от нов URL не се намира(самият път се е променил).

**Първият начин** да решим това: да пренасочим път и на CSS файл, който после успешно ще се намира в една и съща директория с HTML файл.

RewriteEngine On

RewriteRule "^(referat)$" "/web/referats/url/$1.html"

RewriteRule "^(referat\.css)$" "/web/referats/url/$1" [L]

**Вторият начин** е: да пренасочим завинаги или временно със съответните флагове [R=301] или [R=302], ако за нас няма разлика, как ще изглежда URL адрес при вече заредена страници. При такъв подход адрес на заредена страница ще съответства на абсолютен път до самият ресурс (Патерн част на RewriteRule ще се вижда преди страницата да се е заредила, а вече после част Заместване ще се вижда в адресното поле отгоре).

RewriteEngine On

RewriteRule "^(referat)$" "/web/referats/url/$1.html" [R=302,L]

## URL rewriting: кога създава затруднения

Освен предимствата на техниката URL rewriting като улеснено запомняне и обработка на адреси от търсачки и използването на същият URL адрес дори след смяна на технология, то пак има и лек недостатък.

Да предположим, че след пренаписването на URL адрес той изглежда така:

http://www.example.com/Blog/06/04/02/

Сега ако потребител иска ръчно да промени URL адрес, че да достъпи други секции на УЕБ сайта примерно, то може да се сблъсне с недостатъка на описание. Параметрите на заявка липсва(query parameters), а по-точно техните имена. Така може трудно да бъде да се разпознае дата.

В такава ситуации първоначален формат на URL адрес би бил по-подходящ:

http://www.example.com/Blog/Posts.php?Year=06&Month=04&Day=02

## Използвани ресурси:

[1] Paul Tero, “Introduction To URL Rewriting”, зададен/публикуван 02.11.2011, [<http://coding.smashingmagazine.com/2011/11/02/introduction-to-url-rewriting/>], последно посетен на 10.04.2024  
[2] Computer Hope, “URL”, зададен/публикуван 03.05.2023, [<http://www.computerhope.com/jargon/u/url.htm>], последно посетен на 10.04.2024  
[3] Jessica Wilkins, “Mailto Link – How to Make an HTML Email Link [Example Code]”, зададен/публикуван 16.11.2021, [<https://www.freecodecamp.org/news/mailto-link-how-to-make-an-html-email-link-example-code/>], последно посетен на 10.04.2024   
[4] Kai Yuan, “Jdbc URL Format for Different Databases”, зададен/публикуван 08.01.2024, [<https://www.baeldung.com/java-jdbc-url-format>], последно посетен на 10.04.2024   
[5] “Rewrite engine”, зададен/публикуван 05.03.2024, [<http://en.wikipedia.org/wiki/Rewrite_engine>], последно посетен на 10.04.2024  
[6] Oracle, “What Is URL”, [<https://docs.oracle.com/javase/tutorial/networking/urls/definition.html>], последно посетен на 10.04.2024  
[7] Apache, “Apache Module mod\_rewrite”, зададен/публикуван 19.10.2023, [<https://httpd.apache.org/docs/current/mod/mod_rewrite.html>], последно посетен на 10.04.2024

[8] Dave Child, “URL Rewriting for Beginners”, зададен/публикуван 04.08.2008, [<https://aloneonahill.com/blog/url-rewriting-for-beginners/>], последно посетен на 10.04.2024   
[9] Apache, “Apache mod\_rewrite Introduction”, зададен/публикуван 19.10.2023, [<http://httpd.apache.org/docs/current/rewrite/intro.html>], последно посетен на 10.04.2024  
[10] “URL”, зададен/публикуван 24.02.2024, [<https://en.wikipedia.org/wiki/URL>], последно посетен на 10.04.2024

[11] “What is a URL”, [<https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL>], последно посетен на 10.04.2024  
[12] “How to Access an FTP Server from the Browser”, зададен/публикуван 10.05.2022, [<https://www.exavault.com/blog/how-to-access-ftp-server-from-browser>], последно посетен на 10.04.2024

[13] “Clean URL”, зададен/публикуван 20.03.2024, [<https://en.wikipedia.org/wiki/Clean_URL>], последно посетен на 10.04.2024

[14] “Perl Compatible Regular Expressions”, зададен/публикуван 12.03.2024, [<https://en.wikipedia.org/wiki/Perl_Compatible_Regular_Expressions>], последно посетен на 10.04.2024

[15] “Update Apache Configuration”, зададен/публикуван 11.08.2023, [<https://gcore.com/learning/how-enable-apache-mod-rewrite/#3-update-apache-configuration>], последно посетен на 10.04.2024

[16] Ralf S. Engelschall, “URL Rewriting Guide”, зададен/публикуван 1997, [<https://httpd.apache.org/docs/2.0/misc/rewriteguide.html>], последно посетен на 10.04.2024

[17] “Uniform Resource Identifier”, зададен/публикуван 09.04.2024, [<https://en.wikipedia.org/wiki/Uniform_Resource_Identifier>], последно посетен на 10.04.2024

[18] “Create, read, update and delete”, зададен/публикуван 26.10.2023, [<https://en.wikipedia.org/wiki/Create,_read,_update_and_delete>], последно посетен на 10.04.2024

## Използвани снимки:

[1] Фигура-1: пълна сигнатура на URL адрес заедно с незадължителни параметри.

[2] Фигура-2: скриншот за необходимите промени в конфигурационен файл httpd.conf за конкретна директория.

[3] Фигура-3: скриншот от браузъра след въвеждане в адресното поле на пълния път до HTML файла, който се пази локално, и изпълняването му.

[4] Фигура-4: скриншот от браузъра след въвеждане в адресното поле на съкратения URL адрес до HTML файла, който с помощта на Apache mod\_rewrite пренасочва потребителя до необходим ресурс.