

# URI — сложно о простом (Часть 1)

Зиновьев Антон :: 12.07.2015

---

## Uniform Resource Identifier

Привет хабр!

Появилось таки некоторое количество времени, и я решил написать сий пост, идея которого возникла уже давно.

Связан он будет с такой, казалось бы, простой вещью, как URI, детальному рассмотрению которой в рунете уделяется как-то мало внимания.

"Пфф, ссылки они и в Африке ссылки, чего тут разбираться?" — скажете вы, тогда я задам вопрос:

*Что есть что и куда нас приведет?*

- `http://example.com`
- `www.example.com`
- `//www.example.com`
- `mailto:user@example.com`

Если вы не знаете однозначного ответа или вам просто интересно ~~и если вы не боитесь огромного количества трехбуквенных аббревиатур~~ — милости прошу под кат.

Перед тем как начать хотел бы обозначить, что есть [пост](#) на схожую тему, в котором все обозначено проще и немного понятнее. Целью же этого поста, я ставлю более глубокое изучение вопроса и сбор информации об URI в одном месте, дабы «не потерять». Ну, почти в одном месте, статья будет разделена на две части

А для удобства бахнем оглавление, которое работает не без особенностей URI, которую мы рассмотрим попозже, в этой статье.

### ОГЛАВЛЕНИЕ

## Ознакомление

# 1. URI

Унифицированный Идентификатор Ресурса, в простонародье — **URI**

Самое свежее описание того, чем же все-таки являются эти пресловутые URI датируется январем аж 2005-го, а именно [RFC3986](#), написанный самим Тимом Бёнесом-Ли, родоначальника всеми нами любимого *тырнета*.

Резюмируя п.1.1 можно сформулировать определение:

**URI** — *последовательность символов, идентифицирующая физический или абстрактный ресурс, который не обязательно должен быть доступен через сеть Интернет, причем, тип ресурса, к которому будет получен доступ, определяется контекстом и/или механизмом. Например:*

- перейдя по `http://example.com` — мы попадем на http-сервер ресурса идентифицируемого как `example.com`
- в то же время `ftp://example.com` — приведет нас на ftp-сервер того же самого ресурса
- или например `http://localhost/` — URI идентифицирующий саму машину откуда производится доступ

В современном интернете, чаще всего используется две разновидности URI — URL и URN.

Основное различие между ними — в задачах:

- **URL** — *Uniform Resource Locator*, помогает найти какой либо ресурс
- **URN** — *Uniform Resource Name*, помогает этот ресурс идентифицировать

Упрощая: URL — отвечает на вопрос: «Где и как найти что-то?», URN — отвечает на вопрос: «Как это что-то идентифицировать».

## Парочка интересностей про URI

Многие из вас замечали, что на разных ресурсах ссылки называют то URL, то URI и, вероятно, становилось интересно — какой же из вариантов правильный?

Дело в том, что URL увидел свет и был документирован в 1990 году, в то время как URI был документирован лишь в 1994 году. И вплоть до 2002 года, до выхода [RFC3305](#), уместными были оба варианта именования, что, порой вносило путаницу.

В п.2 RFC3305 сообщается об устаревании такого термина как URL, применимо к ссылкам, и что отныне верным будет именование URI, с того момента, во всех документах W3C использует термин URI. Исходя из этого, применяя термин URL к соответствующим ссылкам, вы не делаете смысловой ошибки, но делаете ее с точки зрения правильного именования.

Так же примечателен тот момент, что вплоть до выхода [RFC2396](#), в 1997 году, URI расшифровывался как *Universal Resource Identifier*, что можно увидеть в [RFC1630](#)

Обобщая всевозможные варианты, URI имеет следующий вид:



Забегая вперед, стоит отметить, что не все три компонента являются строго обязательными. Для того чтобы ссылка считалась URI необходимо наличие:

- либо `scheme+authority+path`,
- либо `sheme+path`,
- либо только `path`.

## 1.1. Синтаксис

Согласно п.2 [RFC3986](#):

URI составлен из ограниченного набора символов, состоящих из цифр, букв и нескольких графических символов, все эти символы вписываются в кодировку US-ASCII (ASCII). Зарезервированное подмножество символов может использоваться, чтобы разграничить компоненты синтаксиса в URI, в то время как остающиеся символы: не зарезервированный набор и включая те зарезервированные символы, которые не действуют как разделители в данной компоненте URI, определяют данные идентификации каждого компонента.

### Зарезервированные символы

Зарезервированные символы делятся на два типа:

- *gen-delims*, они же «главные разделители», т.е. символы, разделяющие URI на крупные компоненты.

```
":", "/", "?", "#", "[", "]", "@"
```

- *sub-delims*, они же «под разделители» — символы, которые разделяют текущую крупную компоненту, на более мелкие составляющие, для каждой компоненты они свои, вот список самых распространенных:

```
"! ", "$", "&", "'", "(", ")", "*", "+", ",", ";", "="
```

## Не зарезервированные символы

Исходя из предыдущего пункта, не зарезервированные символы — символы, не входящие в `gen-delims`, а так же не значимые для данной компоненты `sub-delims`. Но в общем случае это:

```
ALPHA, DIGIT, "-", ".", "_", "~"
```

Для данного случая, согласно [ABNF](#):

ALPHA — любая буква верхнего и нижнего регистров кодировки ASCII (в `regExp` `[A-Za-z]`)

DIGIT — любая цифра (в `regExp` `[0-9]`)

HEXDIG — шестнадцатиричная цифра (в `regExp` `[0-9A-F]`)

## Процентное кодирование

В случае, если используются символы выходящие за пределы кодировки ASCII используется механизм т.н. "Процентного кодирования". Так же он применяется для передачи зарезервированных символов в составе данных. Зарезервированные символы, по правилам, не участвуют в процентном кодировании.

Процентно-кодированный символ представляет из себя символьный триплет, состоящий из знака "%" и следующих за ним двух шестнадцатиричных чисел:

```
pct-encoded = "%" HEXDIG HEXDIG
```

Т.о., %20, например, означает пробел.

## 1.2. Компоненты URI

Следующий список содержит описания крупных компонент, составляющих URI:

- **Scheme (схема)**

Каждый URI начинается с имени схемы, которое относится к спецификации для присвоения идентификаторов в этой схеме. Также, синтаксис URI — объединенная и расширяемая система именования, причем, спецификация каждой схемы может далее ограничить синтаксис и семантику идентификаторов, использующих эту схему.

Название схемы обязательно начинается с буквы и далее может быть продолжено любым количеством разрешенных символов.

Разрешенные символы для схемы:

```
ALPHA, DIGIT, "+", "-", "."
```

- **Authority (честно говоря, не знаю как перевести слово на русский, без потери смысла)**

Компонента authority начинается с двойного слеша(//) и заканчивается следующим слешем(/), знаком вопроса(?) или октоторпом(#) (да-да, «решеточка» зовется именно так=) или концом

URI

Authority состоит из:

```
[ userinfo "@" ] host [ ":" port ]
```

где в квадратных скобках опциональные компоненты

Каждую из подкомпонент, отдельно, мы рассмотрим чуть позже, в разделе посвященном URL.

- **Path (Путь)**

Компонента пути содержит данные, обычно, организованные в иерархической форме, которые, вместе с данными в неиерархическом компоненте запроса (Query), служит, чтобы идентифицировать ресурс в рамках схемы URI и authority (если таковая компонента указана). Путь начинается со слеша(/) и заканчивается знаком вопроса(?), октоторпом(#) или концом URI

Разрешенные символы для пути:

```
Не зарезервированные, процентно-кодированные, sub-delims, ":",  
"@"
```

- **Query (Запрос)**

Компонента запроса содержит данные, организованные в неиерархической форме, которые, вместе с данными в иерархическом компоненте пути (Path), служит, чтобы идентифицировать ресурс в рамках схемы URI и authority (если таковая компонента указана). Запрос начинается с первого знака вопроса(?) и заканчивается октоторпом(#) или концом URI

Разрешенные символы для запроса:

```
Не зарезервированные, процентно-кодированные, sub-delims, ":",  
"@", "/", "?"
```

В запросе чаще всего передаются данные в формате key=value (ключ=значение), значение рекомендуется передавать в процентно-кодированном виде, обусловлено это тем, что в значении может встретиться символ "&", который используется для разделения пар ключ-значение, в результате появления такого артефакта дальнейшая последовательность пар ключ-значение может быть нарушена.

- **Fragment (Фрагмент)**

Компонента фрагмент позволяет осуществить косвенную идентификацию вторичного ресурса по отношению к первому.

Семантика фрагмента никак не ограничена, фрагмент начинается октоторпом(#) и заканчивается концом URI, при этом может состоять из абсолютно любого набора символов. Примером применения фрагментов является оглавление данной статьи. Оно состоит из относительных ссылок

```
<a href="#someanchor"></a>
```

а по статье, в определенных местах, раскиданы т.н. «якоря» — теги

```
<anchor>someanchor</anchor>
```

Переходя по указанной в оглавлении ссылке, браузер производит переход ко вторичному ресурсу относительно данной страницы, т.е. скроллит вниз, до появления нужного

```
<anchor>
```

на экране.

На этом, пожалуй, знакомство с URI можно закончить и начать углубляться в отдельные подвиды URI, а именно

## 2. URL

Стандарт URL документирован в [RFC1738](#).

Из п.2:

URL используются, чтобы определить местоположение ресурсов, обеспечивая абстрактную идентификацию расположения ресурса. Определив местоположение ресурса, система может выполнить множество операций на ресурсе, которые могут быть характеризованы такими словами как 'доступ', 'обновление', 'замена', 'поиск атрибутов'. В целом только метод доступа должен быть определен для любой схемы URL.

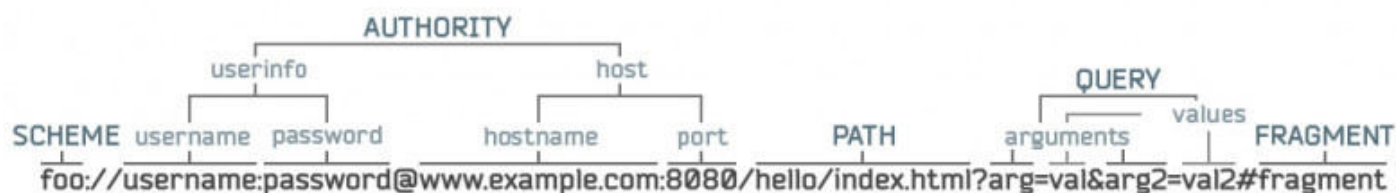
Т. о.: URL призван решить широкий ряд задач, начиная с получения и заканчивая изменением данных на ресурсе, а обязательным параметром для получения доступа — является метод, т. е. любой полноценный (абсолютный) URL можно свести к виду:

```
<scheme>:<часть свойственная этой схеме>
```

### 2.1. Структура

В целом, URL имеет схожую структуру, для всех схем, хотя для каждой отдельно взятой схемы, структура может отличаться от общего шаблона.

Графически ее можно выразить в следующем виде:



И вот, примерно на этом моменте, можно рассмотреть различия между абсолютными (absolute) и относительными (relative) URL, данные определения распространяются не только на URL, но и на URI в целом.

- **Относительная ссылка** использует иерархический синтаксис, чтобы выразить ссылку URI относительно пространства имен другого иерархического URI.

Относительные ссылки так же делятся на несколько подвидов:

- **Ссылка сетевого пути**

Имеет вид:

```
//<authority> <path> [<query>] [<fragment>]
```

Такой тип ссылок применяется не часто, смысл заключается в переходе по указанной ссылке с применением текущей схемы.

Т. е.: находясь, например на `http://example.com` и перейдя по ссылке

`//domain.com` — мы попадем на `http://domain.com`

А в случае если перейдем по той же ссылке с `ftp://example.com`, то попадем на `ftp://domain.com`

- **Ссылка абсолютного пути**

Имеет вид:

```
<path> [<query>] [<fragment>]
```

На этот раз мы останемся в пределах текущего хоста, но попадем по пути `path` в любом случае, по какому бы пути мы сейчас не находились.

Т. е.: даже находясь на `http://example.com/just/some/long/path` и перейдя по ссылке `/path`, мы попадем на `http://example.com/path`

- **Ссылка относительного пути**

Имеет вид:

```
<path> [<query>] [<fragment>]
```

Теперь же, мы будем перемещаться в пределах текущего положения.

Т. е.: находясь на `http://example.com/just/some/long/path` и перейдя по ссылке `path`, мы попадем на `http://example.com/just/some/long/path/path`

- **Ссылка того же документа**

Фактически это ссылки, состоящие только из фрагментарной части URI, либо же

ссылки, у которых все компоненты за исключением фрагментарной совпадают с исходной.

Т.е. `#fragment` и `http://habrahabr.ru/topic/232385/#fragment` являются ссылками того же документа.

- **Абсолютная ссылка** — ссылка вида

```
<scheme> <authority> [<path>] [<query>] [<fragment>]
```

Применяя абсолютные ссылки, мы будем попадать на нужный нам ресурс вне зависимости от того, откуда мы переходим.

Т. е.: находись мы на `http://example.com/just/some/long/path` или же на `ftp://example.com`, перейдя по `http://domain.com/path`, мы в любом случае попадем на `http://domain.com/path`

После того, как мы разобрались с тем, что же такое относительные и абсолютные пути — можно отвечать на вопрос заданный в начале поста:

- <http://example.com> — откроет `http://example.com`
- [www.example.com](http://example.com) — по-идее должен открыть `http://habrahabr.ru/topic/232385/www.example.com`, но хабр сам исправляет ссылку, хотя по стандартам `www.example.com` — ссылка относительного пути
- [//www.example.com](http://example.com) — откроет `www.example.com` со схемой с которой вы просматриваете текущую страницу, т.е. скорее всего будет открыт `http://example.com`
- <mailto:user@example.com> — здесь уже вступают в силу настройки браузера, он предложит вам открыть эту ссылку при помощи почтовой программы и отправить электронное письмо адресату `user@example.com`, а так, это абсолютный URL со схемой `mailto`

Мы уже рассмотрели крупные компоненты, а теперь давайте углубимся в детали построения URL.

- **Scheme** — как говорилось ранее: схема определяет метод доступа к ресурсу. Список актуальных схем можно посмотреть [тут](#).
- **Userinfo** — под-компонент `authority`, использующийся для авторизации пользователя на ресурсе. Состоит из `username` и необязательного `password`, от остальной части `authority` отделяется символом `"@"`. Не смотря на то, что параметр `password` указан в спецификации, его использование крайне не рекомендуется, т. к. фактически производится передача пароля к учетной записи `username`, в незашифрованном виде.

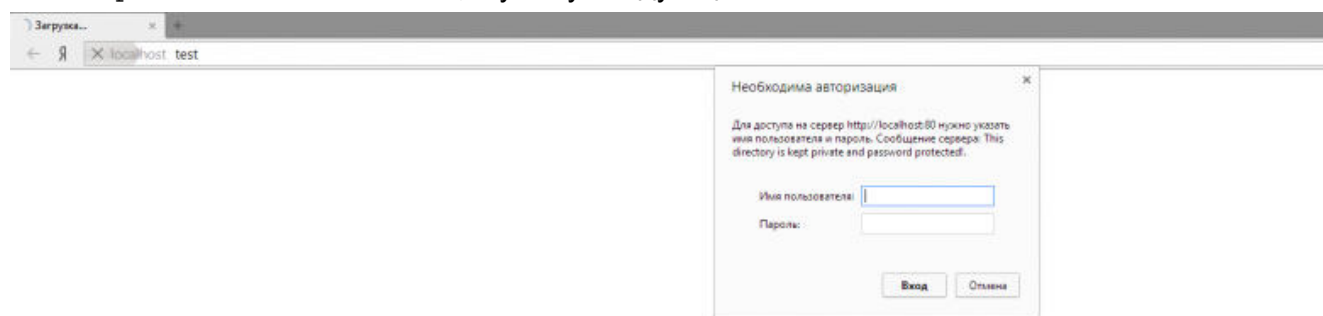
Разрешенные символы:

```
Не зарезервированные, процентно-кодированные, sub-delims, ":"
```

Пример можно привести следующий:



На локалке есть папка test, на которую стоит доступ по паре логин-пароль. То есть, перейдя по `http://localhost/test/`, я увижу следующее:



А если я перейду по ссылке `http://admin:admin@localhost/test/`, то процедура авторизации произойдет автоматически, с данными указанными в блоке `userinfo`:



- **Host** — компонент authority, использующийся для определения целевого узла (или ресурса, если угодно, но понятие «узел» будет более точным), который может находиться как в сети интернет, так и вне её, в зависимости от указанной схемы. Данная компонента не чувствительна к регистру.

Хост может представлять из себя либо IP-адрес, либо регистрационное имя (reg-name) и, опционально, следующий за ними порт(port).

Предусматривается как поддержка существующих форматов IP-адресов (IPv4, IPv6), так и всевозможных будущих, которые будут описаны впоследствии.

Регистрационное имя — привычные нам, т. н. доменные имена — последовательность символов, обычно предназначенных для поиска в локально определенном узле или реестре имени службы, хотя специфичная для схемы семантика URI может потребовать, чтобы вместо этого использовался определенный реестр (или фиксированная таблица имен). Наиболее распространенный механизм реестра имен — Система Доменных Имен (DNS). Доменное имя, используемое для поиска в DNS, состоит из доменных меток, разделенных при помощи ".", каждая доменная метка может содержать следующие символы:

Не зарезервированные, процентно-кодированные, sub-delims

Синтаксис регистрационного имени позволяет использование процентно-кодированных символов, для представления не-ASCII символов, в едином порядке, не зависящем от технологии разрешения имен. Символы, не входящие в ASCII, должны быть сначала закодированы в UTF-8, а затем каждый октет UTF-8 последовательности должен быть процентно закодирован.

В случае, если регистрационное имя с не-ASCII символами представляет собой многоязычное доменное имя, разрешаемое через DNS, оно должно быть преобразовано в

кодировку IDNA ([RFC3490](#)) до поиска имени и, как следствие, регистраторами доменных имен такие регистрационные имена должны предоставляться в кодировке IDNA.

Port (Порт) — десятичный номер порта, отделяется от hostname одним двоеточием ":", может состоять только из цифр. Схема может определять порт по-умолчанию, который будет использоваться в случае если порт не указан. Например, для схемы HTTP порт по-умолчанию — 80, что соответствует зарезервированному под неё порту 80/TCP. Тип порта, так же как и назначенный номер порта, определяется схемой.

- Компоненты Запрос и Фрагмент полностью описаны ранее.

### 3. URN

Стандарт URN документирован в [RFC2141](#).

Из п.1:

Унифицированные имена ресурсов (URN) предназначены, чтобы служить постоянными, независимыми от расположения, идентификаторами ресурсов и разработаны для упрощения отображения других пространств имен (которые совместно используют свойства URN) в URN-пространство. Таким образом, синтаксис URN обеспечивает средство закодировать символьные данные в форме, которая может быть отправлена посредством существующих протоколов, записана при помощи большинства клавиатур, и т.д.

Т. е., в отличие от URL, который ссылается на како-то место, где хранится документ, URN ссылается на сам документ, и при перемещении документа в другое место ссылка не изменится. В силу того, что URN концептуально отличается от URL, то и система разрешения имен у него другая — DDDS, которая преобразует URN в URL, по которым можно найти ресурс/объект или что бы то ни было, на что ссылается URN.

#### 3.1. Структура

URN имеет следующий вид:

```
"urn:" <NID> ":" <NSS>
```

- **«urn:»** — обязательная, регистронезависимая часть URN
- **NID** — Namespace Identifier, данная компонента определяет синтаксическую интерпретацию компоненты NSS.

Минимальная длина — 2 символа, максимальная — 32, разрешенные символы:

латинские буквы, цифры, "-"

NID должен начинаться только с буквы или цифры.

Так же, слово «urn» для NID является зарезервированным, дабы избежать неоднозначности

при определении URN в целом.

Список официально зарегистрированных NID можно посмотреть [тут](#)

- **NSS** — Namespace Specific String, эта компонента служит непосредственно для передачи каких-либо данных.

Разрешенные символы:

```
латинские буквы, цифры, процентно-кодированные, "(", ")", "+",  
",", "-", ".", ":", "=", "@", ";", "$", "_", "!", "*", "'"
```

Зарезервированные символы:

```
"%", "/", "?", "#"
```

Запрещенные символы должны быть процентно-кодированы. Если указанный символ встретится в явном виде, его позиция будет считаться концом URN:

```
октеты 0-32 (0-20 hex), "\", """, "&", "<", ">", "[", "]",  
"^", "`", "{", "|", "}", "~", октеты 127-255 (7F-FF hex)
```

### Самоидентифицирующийся URN

Такие URN содержат в NID название хэш-функции, а в NSS значение хэша, вычисленного для идентифицируемого объекта. Такие ссылки используются в magnet-ссылках и заголовках p2p-сети Gnutella2.

Например, URN из magnet-ссылки с одного торрент-трекера:

```
magnet:?xt=urn:btih:c68abc1ba9b8c7c4bc373862cad1a8c01d69e53d...
```

С теорией все, во второй части рассмотрим, что можно и что нужно делать с URI, если мы их обрабатываем, а именно — нормализация, разбор и т.д.

За сим откланяюсь, спасибо что читали, надеюсь не было скучно, удачи!