HTTP

{ Hypertext Transfer Protocol }

Yahoo! - NCSA Mosaic

File Edit History Manager View Navigate Tools Hotlists Help

Other Doc

http://www.yahoo.com

What's New Check Email YAHOO! My Personalize Help

Yahoo! Auctions
Pokemon, Rolex, 'N Sync

free_email@yahoo.com

Win Free Flowers
FTD.COM

Search advanced search

Shopping - **Auctions** - Yellow Pages - People Search - Maps - Travel - Classifieds - Personals
Games - Chat - **Clubs**
Mail - Calendar - Messenger - **Companion** - My Yahoo! - News - Sports - Weather - TV -

Tue 06/06/00 9:33:04

HTML

HTTP

Speakerinnen*-Liste

Konstantin

https://speakerinnen.org/de/sign_up

SPEAKERINNEN LISTE

Als Speakerin registrieren | Anmelden ENG | DEU

SPEAKERINNEN

E-Mail

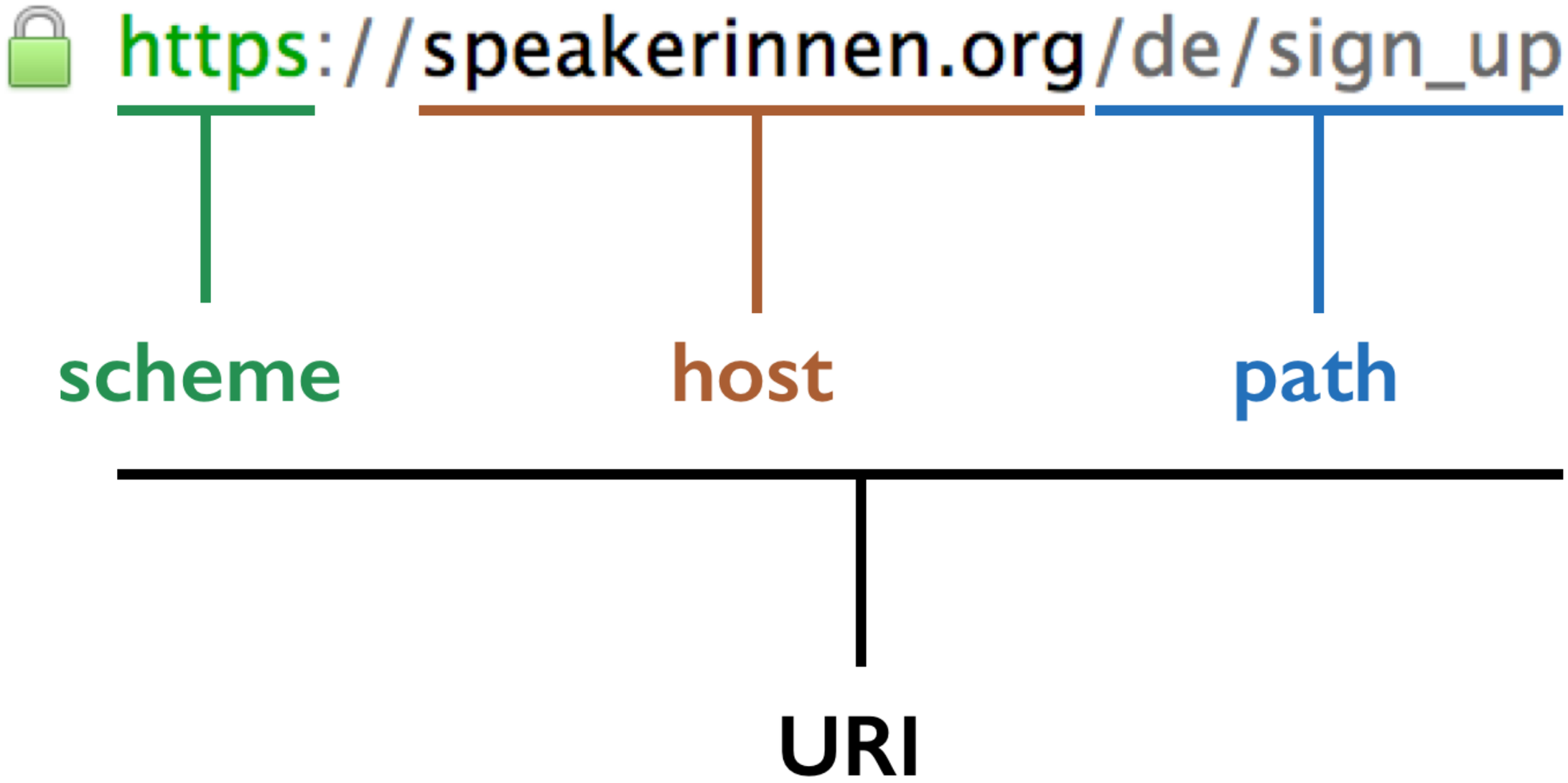
Mindestens 8 Zeichen

Bestätige Dein Passwort

Oder registriere Dich über Twitter

Keine Bestätigungsmail erhalten?

Registrieren


https://speakerinnen.org/de/sign_up
scheme
host
path
URI

`https://speakerinnen.org/de/sign_up`

is translated to

`https://54.255.158.2:443/de/sign_up`

using the DNS protocol

- Connect to the computer reachable under IP address `54.255.158.2`
- Use an SSL/TLS encrypted TCP socket on port `443`
- Speak `HTTP`
- Tell the server we know it as `speakerinnen.org`
- Access resource `/de/sign_up`

Server listens on TCP port

TCP socket

- Is opened when client connects to port
- Client and server can send each other messages

Client starts speaking

```
GET /de/sign_up HTTP/1.1
Host: speakerinnen.org
```

Server replies

```
HTTP/1.1 200 OK
Content-Type: text/html

<html>
  <head>
    <title>Speakerinnen*-Liste</title>
  </head>
</html>
```

request vs response

```
VERB path HTTP/1.1
Header: Value
...

Body (if there is one)
```

```
HTTP/1.1 code description
Header: Value
...

Body (if there is one)
```

Resources

- identified by host and path
- allow multiple operations
- can have multiple representations

speakerinnen.org/de/sign_up

HTTP methods

GET

- Request a resource in its current state.
- The standard operation.
- Request does not include a body.
- Request is safe (and idempotent).

Safe? Idempotent?
Resource state?
What??

Resource state

- Representations, headers and availability associated with a resource.
- `/de/sign_up` exists, is accessible, and has an HTML page with a form as its representation.

Safe requests

- Safe requests do not change the state of a resource.
- HTTP client does not need to ask the user for permission to perform request.

Idempotent requests

- The resource state will be the same after performing the request once or multiple times.
- HTTP client does not need to ask the user for permission to repeat the request.

Non-idempotent requests

- Can't be sure about the resource state after multiple requests.
- HTTP client should always ask the user for confirmation.

localhost:4567

Konstantin

localhost:4567

404

Confirm Form Resubmission

The page that you're looking for used information that you entered.
Returning to that page might cause any action you took to be repeated.
Do you want to continue?

Cancel | **Continue**

GET

- Request a resource in its current state.
- The standard operation.
- Request does not include a body.
- Request is safe (and idempotent).

HEAD

- Same as `GET`, but there wont be a response body.
- Useful if you only care about the headers.
- Request is safe (and idempotent).

POST

- ”Do something dangerous.”
- Default for requests that change something.
- Used for creating a new speakerinnen.
- Unsafe and non-idempotent.

```
<!-- in the /de/sign_up HTML representaion -->
<form action="/de/profiles" method="post">
  <input name="profile[email]" />
  <input name="profile[password]" type="password" />
  <input type="submit" value="Registrieren" />
</form>
```

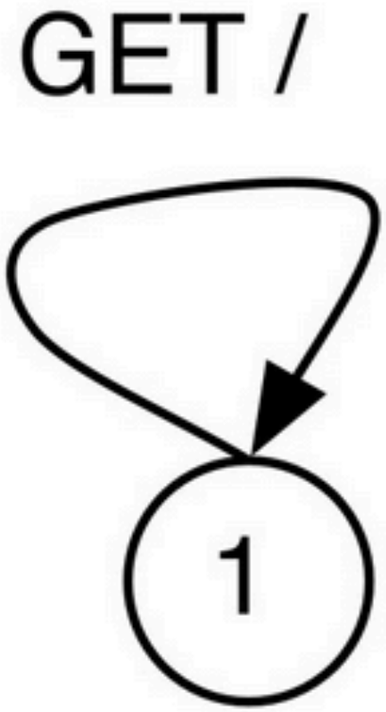
```
POST /de/profiles HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 49

profile[email]=me@rkh.im&profile[password]=abc123
```

PUT

- Writing a resource to a given path.
- Often used for uploads.
- Unsafe, but idempotent (yay).

Other HTTP methods

- `DELETE`: Remove a resource, idempotent.
- `OPTIONS`: Learn about available methods and representations for a resource, safe.
- `PATCH`: Update a resource from a partial representation, non-idempotent.
- `LINK` and `UNLINK`: Create or destroy relations between resources, idempotent.



Repeatable! :)

No state change! :)

Deterministic! :)

Nordic Ruby 2012: We don't know HTTP

by Konstantin Haase

Published June 15, 2012 in Technology

Response Status

- 1xx - informational
- 2xx - success
- 3xx - redirection
- 4xx - client error
- 5xx - server error

If you don't know status xyz, treat it like x00.

Examples

- 303 See Other
- 403 Forbidden
- 404 File Not Found
- 405 Method Not Allowed
- 418 I'm a Teapot
- 500 Internal Server Error

Headers

Common request headers

- `Host`: Domain name in URI.
- `User-Agent`: Client software used.
- `Referer`[sic]: Page user was on before.

Safari on iPad

User-Agent: Mozilla/5.0 (iPad; U; CPU OS 3_2_1 like Mac OS X; en-us) AppleWebKit/531.21.10 (KHTML, like Gecko) Mobile/7B405


Holy Browser wars Batman!
R

Common response headers

- `Server`: Server software used.
- `Last-Modified`: The last time the resource state has changed.

Representations

The "file type" of the response or request body.

We have seen two so far

- `text/html`
- `application/x-www-form-urlencoded`

Other examples

- `image/gif,image/png,image/jpeg`
- `text/plain,text/css,text/x-script.ruby`
- `application/javascript,application/json`

A resource can have multiple representations

But we need to tell it which one we want

We want a PNG image

```
GET /resource HTTP/1.1
Host: example.org
Accept: image/png
```

Server gives us PNG image

```
HTTP/1.1 200 OK
Content-Type: image/png

[ png data ]
```

We want a PNG image

```
GET /resource HTTP/1.1
Host: example.org
Accept: image/png
```

Resource doesn't have a PNG representation

```
HTTP/1.1 406 Not Acceptable
Content-Type: text/plain

No PNG, sorry.
```

We want any kind of image

```
GET /resource HTTP/1.1
Host: example.org
Accept: image/*
```

Server gives us PNG image

```
HTTP/1.1 200 OK
Content-Type: image/png

[ png data ]
```

We want a PNG or GIF image (but prefer PNG)

```
GET /resource HTTP/1.1
Host: example.org
Accept: image/png; q=1.0,image/gif; q=0.5
```

Server gives us PNG image

```
HTTP/1.1 200 OK
Content-Type: image/png

[ png data ]
```

The same works for language.

```
GET /resource HTTP/1.1
Host: example.org
Accept-Language: en, de
```

Cookies

HTTP is stateless.

Cookies are a way to attach state.

```
HTTP/1.1 200 OK
Set-Cookie: mycookie=foobar; HttpOnly
Content-Type: text/plain

Just gave you a cookie!
```

```
GET /somepage HTTP/1.1
Host: example.org
Cookie: mycookie=foobar
```

```
get '/' do
  name = session[:name]
  "Hello #{ name }!"
end
```

Death to Cookies

Konstantin Haase
@konstantinhaase

Distill: Death to Cookies by Konstantin Haase

Published August 9, 2013 in Technology

http://w

Thank You!