HTTPの通信

通信の仕組み

- 1対1で行う通信がユニキャスト
- グループの属する複数に受けて行う通信がマルチキャスト
- あるネットワークの全コンピュータに向けて行う通信がブロードキャスト

通信相手の指定

1. インターネットに接続するコンピュータはユニークなアドレス、グローバルIPアドレスが割り当てられる

2. HTTPを使う通信では、相手を特定するためドメイン名を使う。この名前はIPアドレスと対応しているので指定と同じ効果がある。

3. IPアドレスとともにポート番号を指定する必要がある。HTTPだと80番。

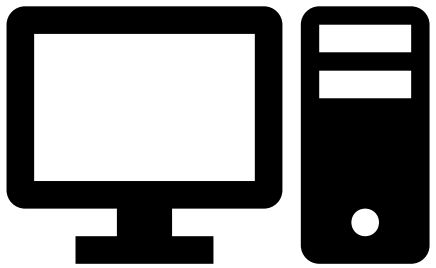
通信の始め方

- HTTPの通信では下位層のプロトコルとして標準にTCPが利用される。

- コネクション型なので3ウェイハンドシェイクが行われる。

HTTPの仕組み

HTTPでは一回のやりとりにできる情報は1つに限られる。
そのためこの動作を繰り返して処理を行なっている



クライアント

(A)サーバにリクエスト

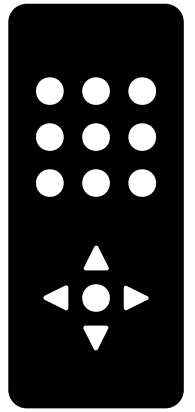
「xxxを取り出して」

(C)レスポンスが返ってくる

目

xxxの中身

リクエストに応じて処理



HTTP　リクエスト

/HTTP/1.1 ← リクエスト行

behiroshi.la.coocan.jp

意味
サーバ上のファイルを取り出す
サーバー上のファイルに関連するヘッダーフィールドだけ取り出す
サーバー上のプログラムを送付する
サーバー上のファイルを書き換える
サーバー上のファイルを削除する
中継のための接続を作る
対象で利用できるメソッドの一覧を読み出す
リクエストをそのままかえす

URLを見てみる

https://www.Yakult.co.up

ホスト　ドメイ

スキーム　URLの先端部分に記載するルールの総称

ホスト名　特定のネットワークに接続された機器の名前

ドメイン名　/の前までの部分のこと

HTTP/1.1 200 OK

ステータスコード

リクエスト処理を

試みた結果

	意味
	リクエストを受信して
	リクエスト処理に成功
リダイレクト系	リクエストを終了するには、動作が必要
クライアントエラー系	リクエスト構文に問題
サーバーエラー系	リクエストは正常だがサーバ側に

ヘッダーフィールドの構成とフィード名

Date: Thu, 05 Nov 2020 01:30:45 GMT

超高速に開ける阿部寛のホームページHTTPのヘッダを見てみよう

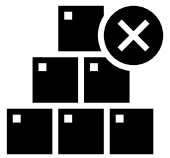
C:¥Windows¥System32¥wsl.exe

```
root@PC-72:/mnt/c/Windows/system32# http get http://abehiroshi.la.coocan.jp/
HTTP/1.1 200 OK
Accept-Ranges: bytes
Connection: keep-alive
Content-Length: 538
Content-Type: text/html
Date: Thu, 05 Nov 2020 01:40:29 GMT
ETag: "21a-59f8969af8600"
Last-Modified: Thu, 27 Feb 2020 07:10:16 GMT
Server: Apache

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
<meta name="GENERATOR" content="JustSystems Homepage Builder Version 20.0.6.0 for Windows">
<meta http-equiv="Content-Style-Type" content="text/css">
<title>･ｃ･･･ﾟ ･ﾙ･ｚ･[･･･ｙ･[･Ｗ</title>
</head>
<frameset cols=18,82>
  <frame src="menu.htm" marginheight="0" marginwidth="0" scrolling="auto" name="left">
  <frame src="top.htm" marginheight="0" marginwidth="0" scrolling="auto" name="right">
  <noframes>
  <body></body>
  </noframes>
</frameset>
</html>

root@PC-72:/mnt/c/Windows/system32# _
```

ヘッダーフィールド

フィールド名	意味
Accept-Ranges	部分的なリクエストに対して対応していることを周知するマーカー
Connection	接続状態に関する通知
Content-Length	内容のサイズ
Content-Type	内容のメディアタイプ
Date	生成した日時
Etag	内容を要約する情報
Last-Modfiled	内容の最終更新日
Server	サーバーのプログラムや名称

HTTPの弱点



- 平文通信するため通信が暗号化されていない
- 相手を確認しない
- 安全性を証明できない

HTTP

- サーバーに対して誰にでもリクエストができる
- 相手が成り済ましたクライアントの可能性もある
- TCP/IP通信は盗聴可能なネットワーク

HTTPS 

- 通信を暗号化する技術

- SSL（Secure Socket Layer）を組み合わせた技術

- SSLによって安全な通信路を確立してから通信を行う

SSLサーバー証明書

- Webサイトの身元の証明やSSLにより通信の暗号化に使われるデジタル証明書

- 証明書は第三者によって発行される

- SSLサーバ証明書があれば安全というわけではない

なぜ全て暗号化にしないのか

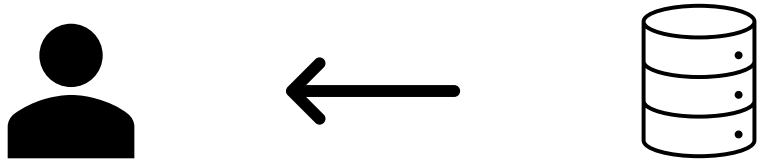
- 暗号化をするとリソースを喰う
- 2倍から100倍の負荷がかかる
- SSL証明書の更新する必要がある

認証とは？

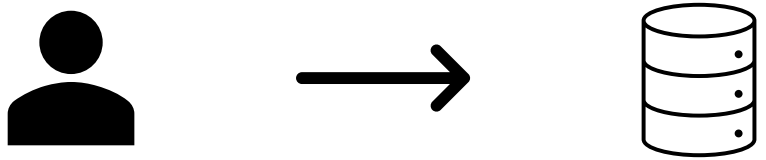
WEBへのアクセスに際してユーザー名とパスワードを求め、それがあらかじめ登録してあるものと合致したときだけ、リクエストされた内容を送り返す仕組み

認証に使われるヘッダ

- WWW-Authenticate
  アクセスに必要な認証の種類などを知らせる
  典型例：WWW-Authenticate:Basic realm="Web access"



- Authorization
  認証の種類やユーザー名/パスワード情報を知らせる
  典型例：Authorization:Basic YwI0MjM0NUBh==



WWW-Authenticateヘッダー

- 機能　アクセスに必要な種類などを知らせる

- 典型例
- WWW-Authenticate: Basic realm=“Web access”

- 認証の種類　digest認証もある

- 認証に必要な情報

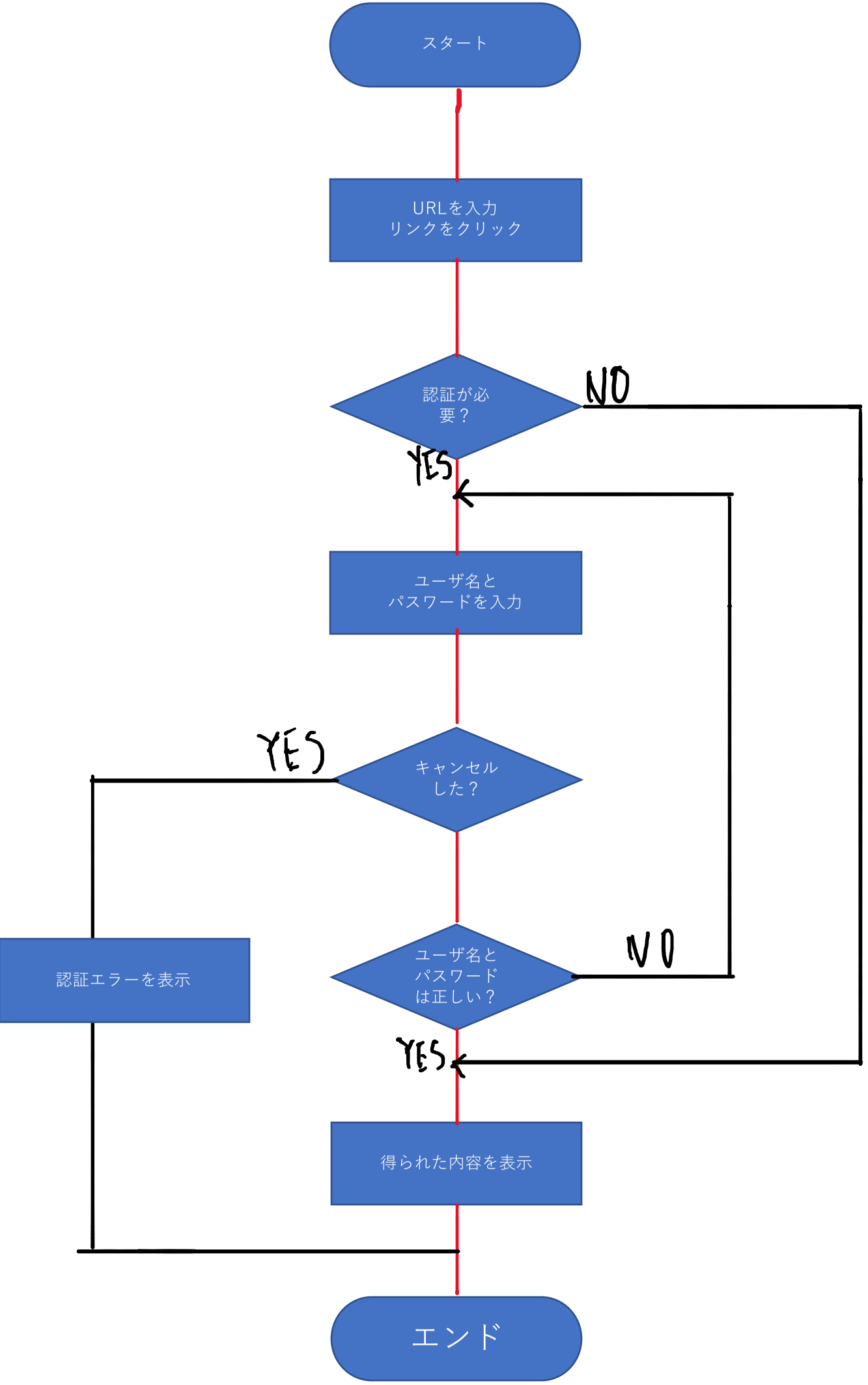
Authorization ヘッダー

- 機能　認証の種類やユーザー名　パスワード情報を知らせる

- 典型例:
- Authorization: Basic Ywlajsiimwnajiaosk==

- 認証の情報

- ユーザー名とパスワードから生成した認証情報


スタート
URLを入力
リンクをクリック
認証が必
要？
NO
YES
ユーザ名と
パスワードを入力
YES
キャンセル
した？
認証エラーを表示
ユーザ名と
パスワード
は正しい？
NO
YES
得られた内容を表示
エンド