4.1 URIの重要性

URI (Uniform Resource Identifier)

同じルールに則って付けた、リソースを識別するIDのこと

URIを使うと、Web上に存在するすべてのリソースを一意にできる

4.2 URIの構文

簡単なURIの例

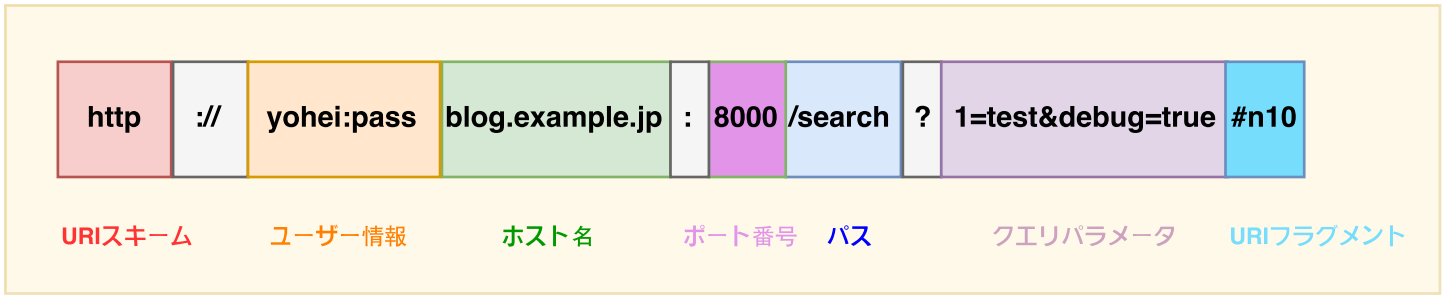
URIスキーム　ホスト名　パス

URIスキーム　そのURIが利用するプロトコルを示すのが一般的
URIスキームと、後続の記述は、「://」で区切られる

ホスト名　DNSで名前解決できる、ドメイン名 or IPアドレス
インターネット上で、必ず一意になる

パス　ホストの中で、リソースを一意に指し示す

複雑なURIの例

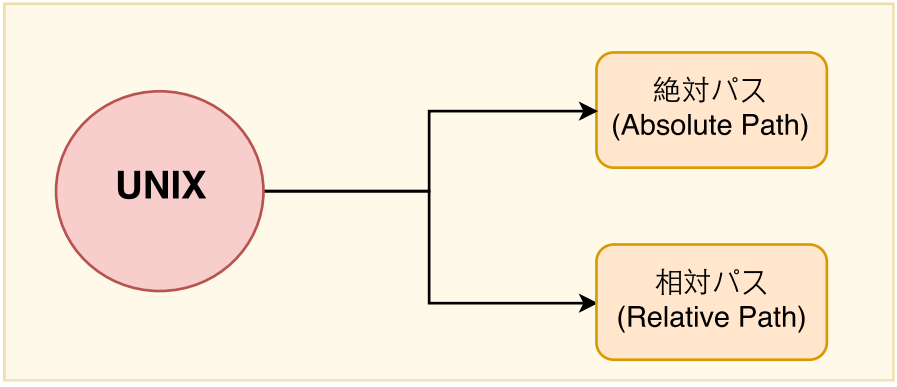


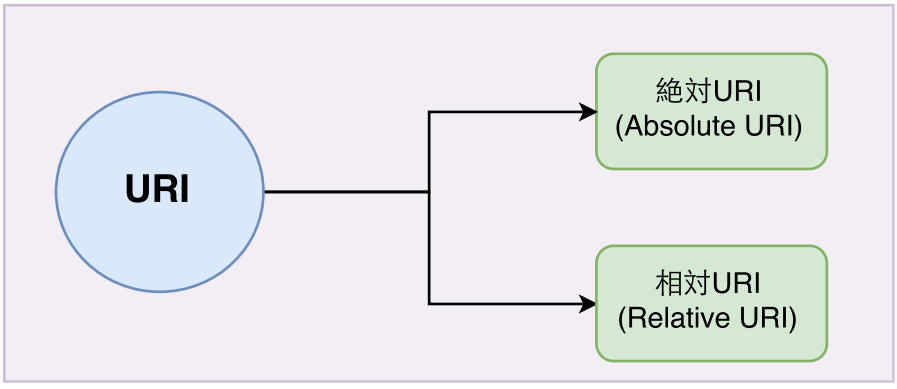
ユーザー情報	リソースにアクセスする際に利用する、ユーザー名とパスワード ユーザー名とパスワードの間は、「:」で区切る
ポート番号	ホストへのアクセス時に利用する、TCPのポート番号 省略した時はデフォルト値が用いられる (e.g. HTTPなら、80)
クエリパラメータ (Query String)	クライアントから動的にURIを生成する時に利用する 「?」の区切り文字の後に、「名前=値」形式で記述する 複数の「名前=値」がある場合は、「&」で連結する
URIフラグメント	URIフラグメントより前までの文字列で表すURIの、 さらに細かいリソース内部を表現する際に用いる

4.3 絶対URIと相対URI

URIのパス

UNIXのファイルシステムと同じような階層構造を持っている





相対URI

相対URIは、ベースURI (起点となるURI) がなければ、クライアントが解釈できない

ベースURIの指定方法は、2つある

リソースのURIをベースURIとする

メリット　：直感的に操作できる
デメリット：リソースのURIを、クライアント側で保存する必要がある

ベースURIを明示的に指定する方法

HTMLやXMLの中で、明示的に指定する (<base> 要素など)
メリット・デメリットは、「リソースのURIをベースURIにする」と逆

4.4 URIと文字

URIで使用できる文字

ASCII文字 (American Standard Code for Information Interchange character) が使用できる

URIのASCII文字は、大文字小文字を区別する

日本語など、ASCII以外の文字をURIに使用する場合はは、「%エンコーディング」方式を用いる

%エンコーディング

%エンコーディングの文字は、大文字小文字を区別しない

エンコーディングに使用するcharsetは、URIを提供するサーバー側で決める
現在は、UTF-8が一般的

クライアント側でフォームを使ってURIを生成する場合には注意が必要
一般的には元になるフォームを提供しているWebページのcharsetを使って解決する

4.5 URIの長さ制限

事実上の制限がある

仕様上はURIの長さに制限はない

IEでは、2038バイトの制限があるため、事実上はこの長さに合わせることになる
なお、Edgeでもほぼ同じ制限がある

4.6 さまざまなスキーム

URIスキームの種類

最初にhttpスキームが誕生してから、さまざまなプロトコルに対応したスキームが作成された

URIスキームは、IANAによって管理されている

4.7 URIの実装で気をつけること

注意すべき2つのこと

相対URIの解決

クライアント側での相対URIの解決は面倒な処理になるので、
なるべく絶対URIを使ったほうがいい

%エンコーディングの扱い

URIにASCII文字以外を入れる場合は、文字エンコーディングの混乱を避けるため、
UTF-8を用いるほうがいい