

5.1 クールなURIは変わらない

Cool URIが必要になる背景

以前アクセスできたサイトが、URIの変更によって、ある日アクセスできない現象が多発した。

Webは、リンクによって実現する、ハイパーメディアシステムである。リンクが切れてしまうことは、パイパーメディアシステムが機能しないことになる。

5.2 URIを変わりにくくするためには

URIの設計指針

- ・プログラミング言語依存の拡張子を利用しない (.pl, .rb, .do など)
- ・実装依存のパス名を利用しない (cgi-bin, servlet など)
- ・プログラミング言語のメソッド名を利用しない (show など)
- ・セッションIDを含めない
- ・URIはそのリソースを表現する名詞である

5.3 URIのユーザビリティ

万人に理解しやすいURI

- ・ 文字数が短いため、覚えやすい
- ・ 開発者以外に馴染みの薄い単語が出てこない (servlet など)

5.4 URIを変更したいとき

Cool URI = 変わらないURI

現在運用しているシステムのURIを安易に変更することは、Cool URIの定義から外れてしまう。

変更したいときはリダイレクトを使う。

`http://example.jp/old`

301
Moved Permanently



`http://example.jp/new`

5.5 URI設計のテクニック

拡張子で表現を指定する

リソースの表現を指定する使い方



マトリックスURI

階層構造で管理できないリソースを表現する際に使用される (地図情報 など)

マトリックスURIの表現方法は2つある

- ・ セミコロン形式 (パラメータの順序に意味がない。key=valueのような形)

`http://example.jp/map/lat=35.705471;lng139.751898`

- ・ カンマ形式 (パラメータの順序に意味がある)

`http://example.jp/map/35.705471,139.751898`

5.6 URIの不透明性

クライアント側の開発は、不透明なURIを利用する

クライアントは、あくまでサーバーが提供するURIをそのまま扱うだけである。

そのため、次のことをしてしまうと、サーバー側でURIが変更されると、システムが動かなくなる可能性がある。

- ・ URIの内部構造を推測して、操作をする
- ・ クライアント側でURIを構築する

refs: <http://shindolog.hatenablog.com/entry/2014/05/27/234944>

5.7 URIを強く意識する

URIは、Webサービス設計において最も重視すべきパーツ

意識しておくポイントが、3つある

- ・ URIはリソースの名前である
- ・ URIは寿命が長い
- ・ URIはブラウザがアドレス欄に表示する