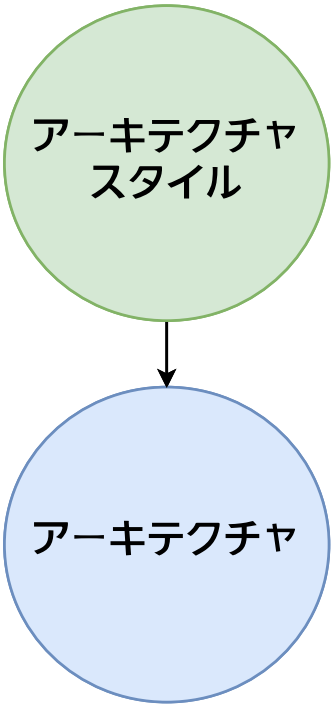
3.1 アーキテクチャスタイルの重要性

アーキテクチャスタイル・アーキテクチャ



アーキテクチャスタイル

アーキテクチャ設計の指針・作法

別名: (マクロ)アーキテクチャパターン

e.g.)

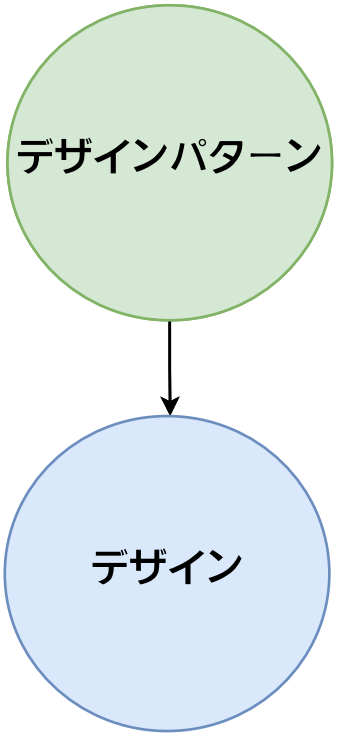
- REST
- Client Server
- MVC

アーキテクチャ

実際のシステムの特性や構造

---

cf) デザインパターン・デザイン



デザインパターン

ソフトウエア設計の指針・作法

別名: マイクロアーキテクチャパターン

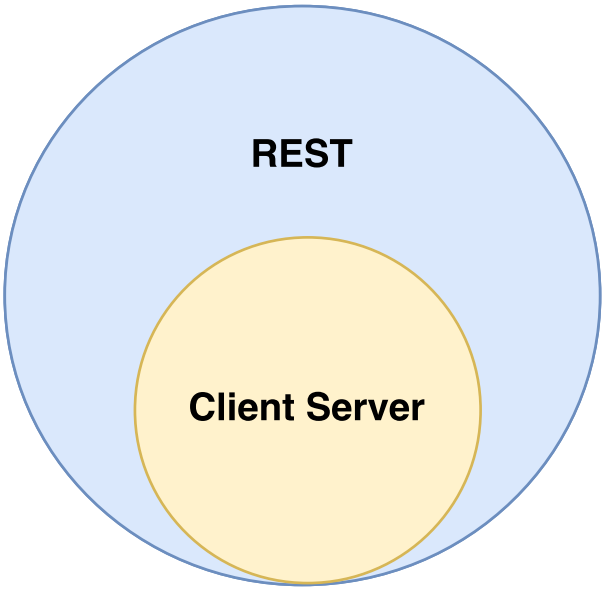
アーキテクチャスタイルよりも、
粒度の小さいクラスなどの設計様式

デザイン

ソフトウエアの設計

3.2 アーキテクチャスタイルとしてのREST

ネットワークシステムのアーキテクチャスタイル



REST

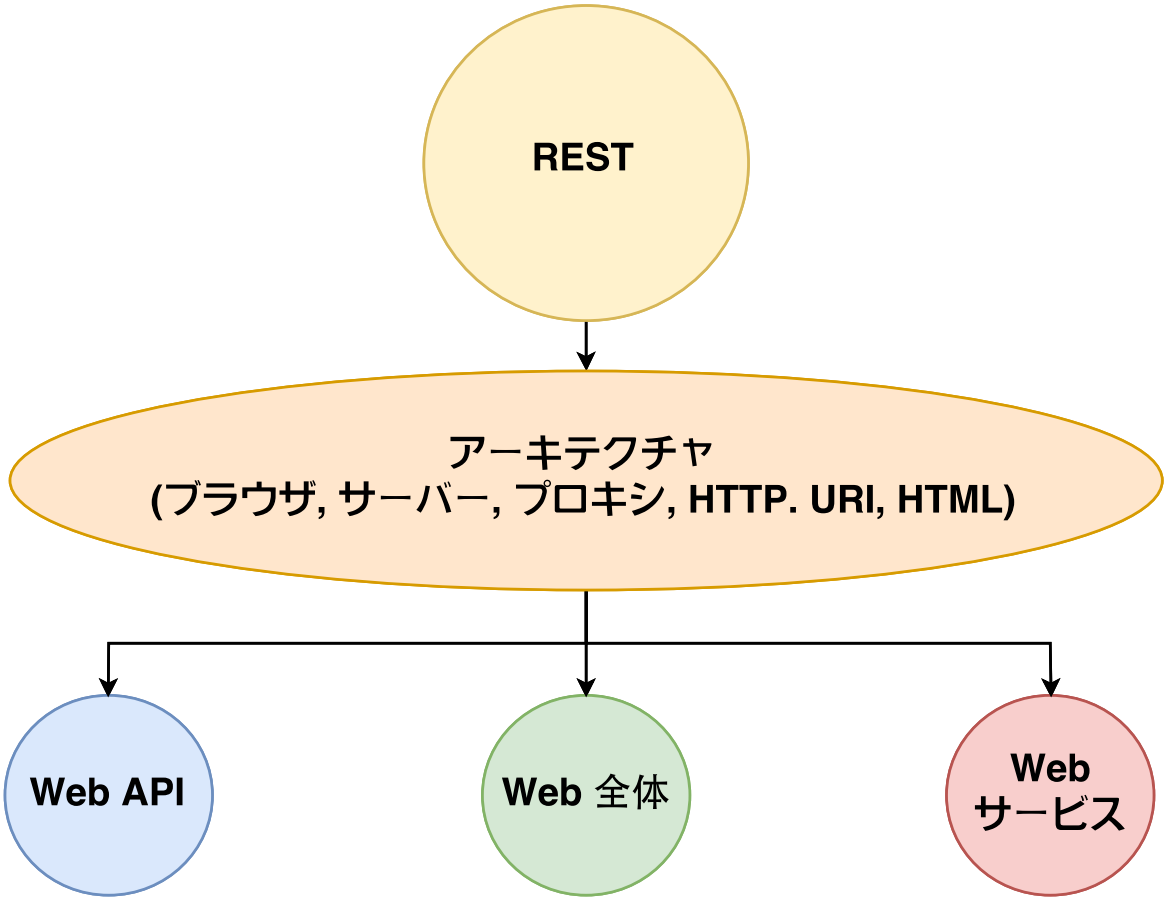
Client Serverに、
いくつかのアーキテクチャスタイルを加えた
ネットワークのアーキテクチャスタイル

Client Server

Webのアーキテクチャスタイル

RESTとWebの実装

アーキテクチャスタイル(REST)を必ず守る必要はないが、守ったほうがWebらしい実装ができる



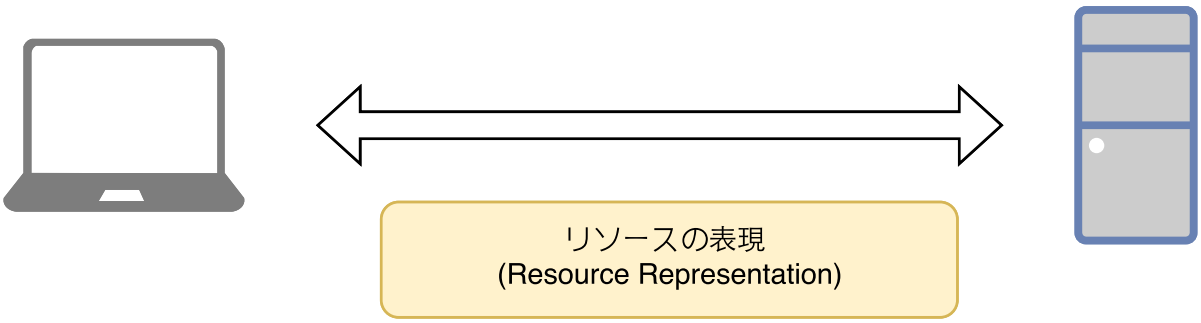
3.3 リソース

リソースとは

Web上に存在する情報

複数の表現を持つことができる (e.g. HTML, PDF, 画像)

状態を持っている



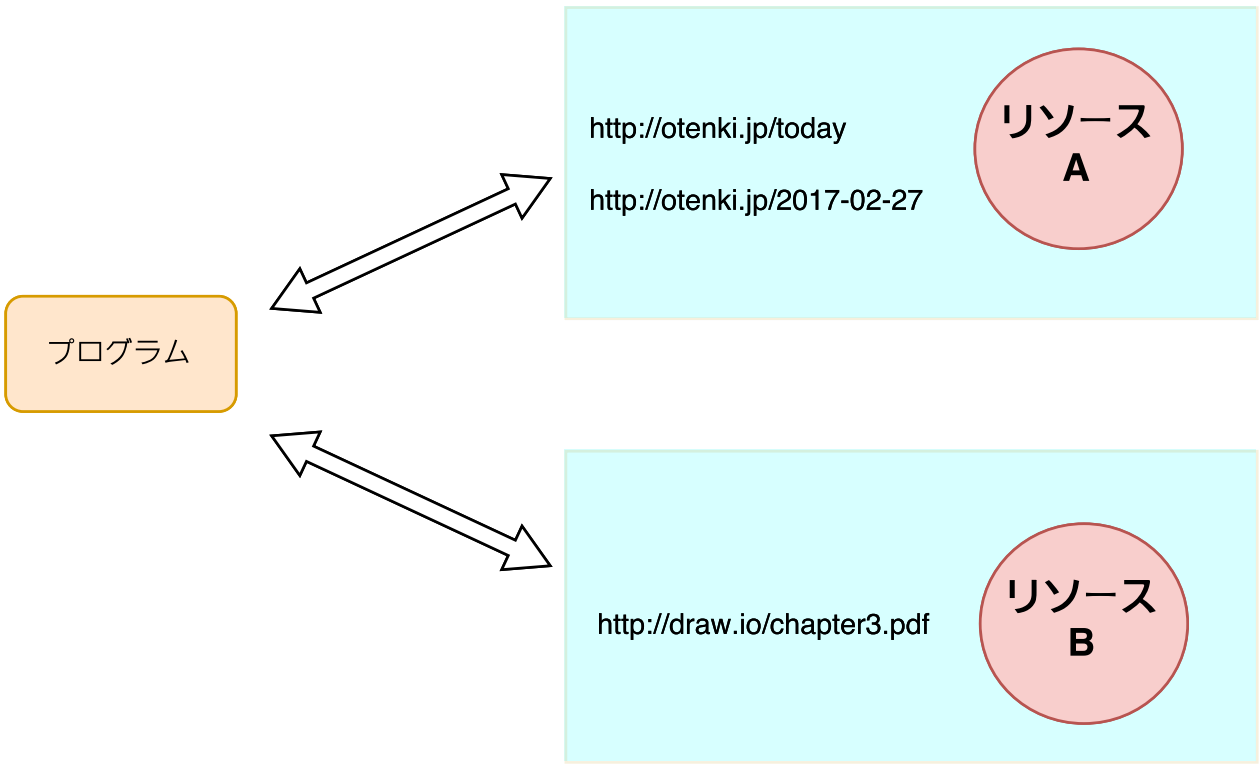
リソースの状態は、時間経過などによって変化する

URI

世界中のリソースにに付ける、一意の名前のこと

プログラムが解釈しやすい名前である

一つのリソースに対して、複数のURIを割り当てることも可能
(ただし、正式なURIがわかりにくくなってしまうデメリットもある)



3.4 スタイルを組み合わせてRESTを構成する

REST = ULCODC$SS

RESTは、6つのアーキテクチャスタイルを組み合わせた、複合アーキテクチャスタイルである

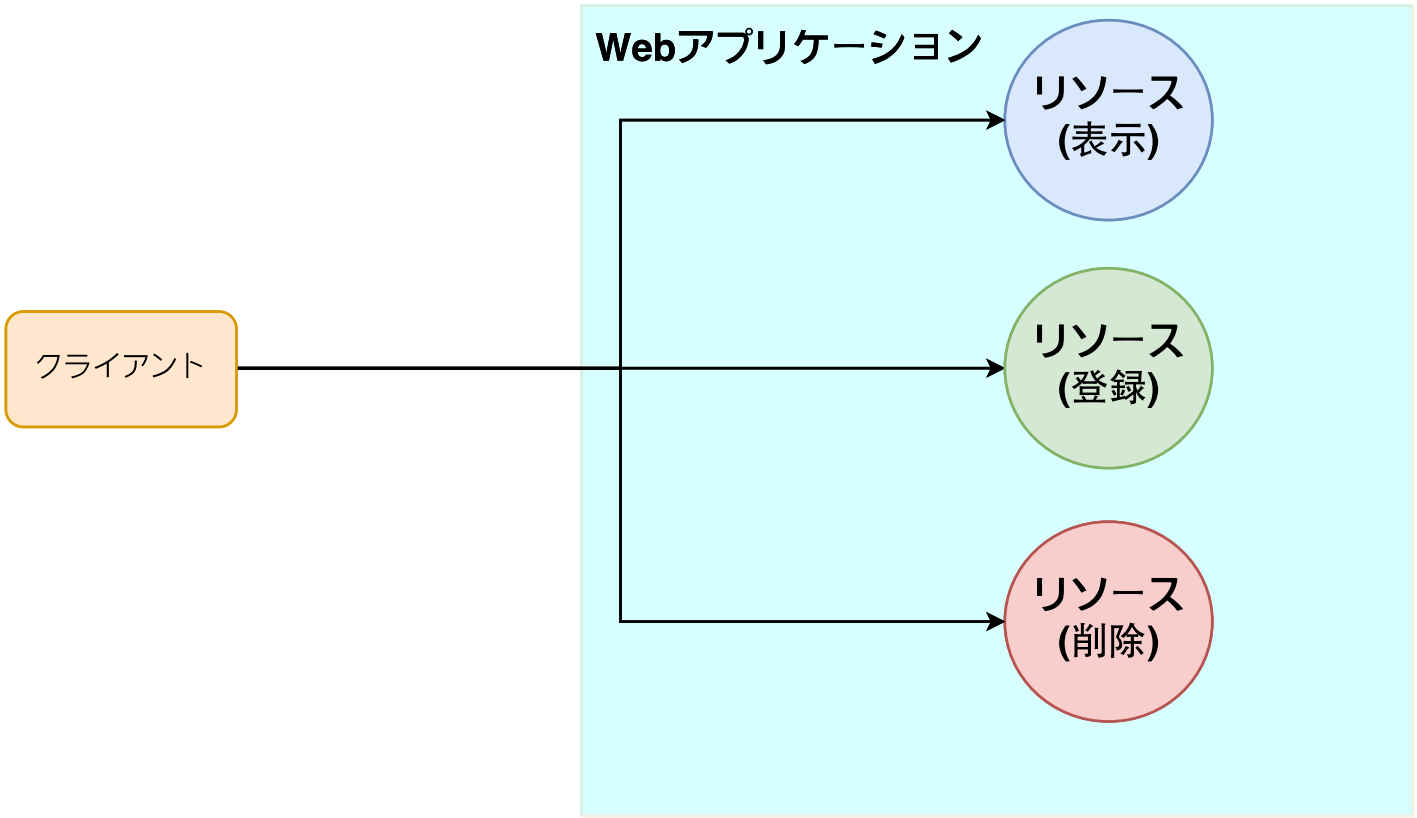
6つのアーキテクチャスタイルは、それぞれの頭文字を取って、**ULCODC$SS**と呼ばれる

- Uniform: 統一インターフエース
- Layered: 階層化システム
- Code on Demand: コードオンデマンド
- $(Cache): キャッシュ (現金のcachと同じ発音のため、$)
- Stateless: ステートレス
- Client Server: クライアント / サーバー

3.5 RESTの2つの側面

RESTとハイパーメディア

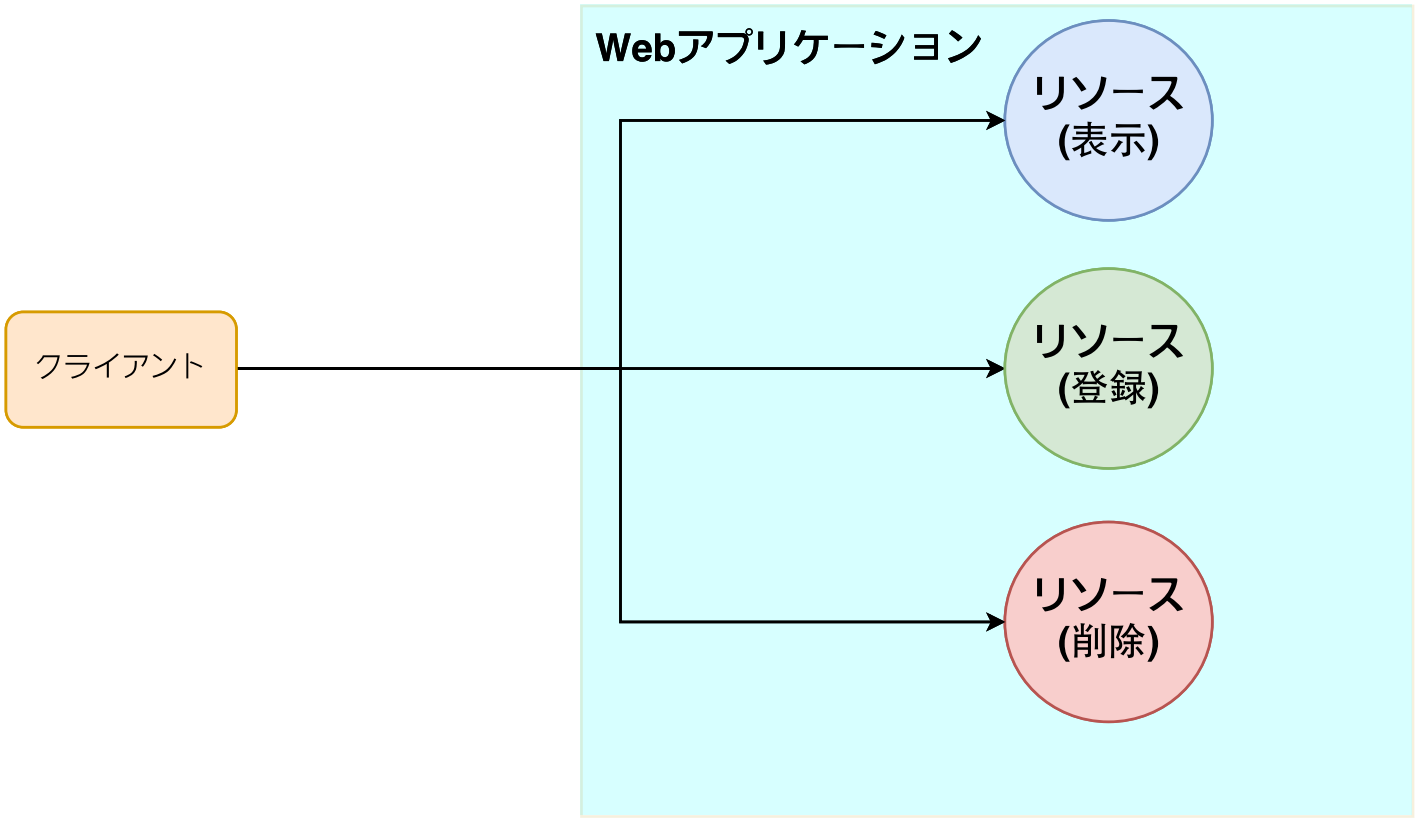
アプリケーション状態エンジンとしてのハイパーメディア



RESTと分散システム

リソース単位でのデータのやりとりは、RPCより粒度が大きいため、性能劣化を抑えられる

リソースに適用できるHTTPメソッドは固定されているので、HTTPを実装したクライアントであれば、インターフェイスを考慮しなくても接続できる



3.6 RESTの意義

Webを良くするには?　-> すべてのWebに関するものをRESTfulにする

RESTfulとは、REST規約に従っている実装のこと