Engineer

冪等

何度実行しても、意図した効果が一度の実行と同じであること

どのようなものか

ある操作を何度実行しても、一度実行したのと効果が同じなら、冪等であるといいます。この性質を冪等性（idempotence、idempotency）と呼びます。数学に由来する言葉で、関数をその結果へ再び適用しても変わらず、f(f(x))がf(x)と等しくなる場合を指します。すでに切り下げた数を、もう一度切り下げても、同じ数になります。

ソフトウェアでは、二回目を試すと分かりやすくなります。値を設定する操作は、二回目も、すでにある値を書くため冪等です。値へ加える操作は、再び加えるため、そうではありません。

```
-- 冪等：2回目も同じ値を設定する
UPDATE accounts SET plan = 'pro' WHERE id = 42;

-- 非冪等：2回目にさらに10を加える
UPDATE accounts SET credits = credits + 10 WHERE id = 42;
```

HTTPが、この言葉を最も厳密に定義します。RFC 9110は、複数の同一要求を送ったときの、意図されたサーバーへの効果が、一回だけ送ったときと同じなら、そのメソッドを冪等とし、PUT、DELETE、安全なメソッドGET、HEAD、OPTIONS、TRACEを挙げます。定義には、重要な細部が二つあります。対象は、クライアントが求めた効果なので、サーバーは、各要求を別々に記録しても構いません。また、応答は変わっても構いません。同じリソースへの二回目のDELETEが、最初の200に対し404を返しても、どちらもリソースがなくなるため、冪等です。PATCHは一覧の外です。RFC 5789は、安全でも冪等でもないとしますが、クライアントは、If-Matchヘッダーへ強いETagを付けて条件付きにすることで、特定のPATCHを冪等に送れます。一回目がリソースを変えた後では、繰り返したコピーを拒否するためです。

二つの約束のうち、安全のほうが厳しいものです。RFC 9110でいう安全なメソッドは、本質的に読み取り専用です。そのため、すべての安全なメソッドは冪等ですが、DELETEは、安全でなくても冪等です。

概念を判断に生かす

「冪等とは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 月末に誰かが表計算ファイルをコピーしなくても、決まった時刻に必要な場所へ数値が届く仕組みをつくります。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 14部門を支える共通インフラ計画 — 14 共通インフラ計画の対象となった研究部門

14部門を支える共通インフラ計画 お問い合わせ →

なぜ重要なのか

ネットワークは、応答を失います。クライアントが要求を送り、応答前に接続が切れると、サーバーが要求を一度も見ていないのか、見て止まったのか、実行して返答を失ったのか、区別できません。自然な復旧は再送であり、それを無害にするのが冪等性です。RFC 9110は、まさにそこへ再試行規則を置きます。通信障害後、冪等な要求は自動で繰り返せます。クライアントは、冪等でない要求を自動で再試行するべきではありません。ただし、実際には冪等と分かっているか、元の要求が一度も適用されていないと判断できる場合は除きます。プロキシーは、自らそのような再試行をしてはなりません。

キューでは、同じ問いが大規模に起きます。多くのメッセージシステムが約束するのは、少なくとも一回の配信です。処理を済ませ、受領確認を返す前に処理側がクラッシュすると、同じメッセージを再び渡されます。ネットワーク越しに厳密に一回だけ配信するのは、はるかに難しい約束なので、実用上の答えは、二回目の配信でも変化しない処理側を作ることです。構成管理も、この性質に依存します。サーバーを宣言した状態へ整えるツールは、変更のたびに、しばしば定期的にも、繰り返し動きます。各回は、すでに正しいサーバーを、見つけたとおりの状態に保つ必要があります。

本当に決めるべきなのは、本来冪等でない操作へ、この性質をどう与えるかです。一般的な答えは三つあります。第一は設計です。「この行を追記する」ではなく、「ファイルにこの行があることを保証する」のように、手順ではなく目標状態で変更を表します。第二はキーです。クライアントが一意な識別子を要求へ付け、サーバーは一意制約の下で記録し、繰り返しには、もう一度実行せず、記録を見つけて元の結果を返します。第三は、動く前に確認する方法です。Ansibleの大半のモジュールは、現在の状態と望む状態を比較し、一致すれば何

もせず終了します。どれにも費用があります。目標状態の設計には、最終状態の明確なモデル、キーには保存領域と保管期間の規則、事前確認には、現在の状態を確実に読む方法が必要です。

実際の活用例

会社自身のサーバーはAnsibleで設定し、そのプロジェクトでは、冪等性を契約とし、収束したホストは変更ゼロを報告します。引数なしのプロビジョニングで、すべてのホストを宣言した状態へ収束させます。すでに正しいホストで、変更ゼロを報告できないプレイは、未完成と扱います。そのため、冪等性は、期待するものではなく、毎回測るものになります。

同じ考えをデータへ適用したものが、十九段階のビルドパイプラインを二回動かすテストです。事例は、対象とする不具合を挙げます。確認せずに挿入する段階や、先行する段階の出力順へ依存する段階によって、初回は動き、二回目で壊す生成ツールです。テストは、空のファイルからデータベースを構築し、スナップショットを取り、その結果に対してパイプライン全体を再実行します。行数、内容、識別子まで、二回目で何も変わらないことを求めます。このテストは、対象とする入力とデータベースの状態について、繰り返し実行した結果の証拠を示します。

限られた在庫の販売では、最後のチケットを一回だけ売るという記事が、キーと条件付き書き込みを組み合わせます。各購入手続きに冪等性キーを持たせ、取り置きテーブルに `UNIQUE (tenant_id, idempotency_key)` を置くため、再クリックでは、もう一つ確保せず、すでにある取り置きを返します。タイムアウトと取り消しボタンなどで、解放処理が二回動いても、自身の更新でちょうど一行を変えた場合だけ在庫を戻すので、一個しか戻りません。さらに後の各メッセージ処理側も、注文をキーとして冪等にします。メッセージは少なくとも一回届き、冪等性によって、効果を実質一回にします。

問題が起きるところ

再試行した要求が、二回課金します。 支払い、注文、メールを作るPOSTは、既定では冪等でなく、クライアントやプロキシーが時間切れ後に再試行すると、二つ目を作ります。サーバーが一意制約の下で保存する冪等性キーを付け、同じキーが戻れば、保存した結果を返します。

空の結果を、エラーと読みます。 PostgreSQLでは、`INSERT ... ON CONFLICT DO NOTHING` が、一意制約へ違反する行を省き、`RETURNING` は、実際に挿入、更新した行だけを返します。同じ行の挿入を繰り返せば、何も返らず、それが二回目の

正しい結果です。空を失敗と扱うコードは、無害な再試行をエラーページへ変えてしまいます。すでにある行を検索するべきです。

コマンドタスクが、毎回ホストを変えます。 Ansibleの文書は、大半のモジュールが、すでに望む状態かどうかを確かめる一方、「すべてのプレイブックがこのように動くわけではなく、すべてのモジュールがそう動くわけでもない」と説明します。コマンドは任意で、何をするかAnsibleには分からないため、コマンドを動かすタスクが、よくある例外です。commandモジュールの `creates` は、指定ファイルがすでにあれば省き、 `removes` は、ファイルがまだある間だけ動かし、 `changed_when` は、どの結果を変更と数えるかを定めます。どれもないタスクは、ホストがどんな状態でも、収束処理のたびに再び動きます。

カウンターが、再試行も数えます。 少なくとも一回の配信で、メッセージごとにカウンターを増やす、または通知を送る処理側は、二回届いた一件に対して、その処理を二回行います。効果とともに、同じトランザクションで、メッセージの業務上のキーを保存し、すでに保存したキーは省きます。

二回目が、初回の結果を壊します。 空のデータベースなら正しくても、データがあると、重複挿入、識別子の振り直し、初回が残した順序への依存で、段階が誤動作する場合があります。何もないところから一度だけ動くテストは、それでも通ります。前述のパイプラインテストのように、ジョブ全体を二回動かし、二回目で何も変わらないことを必須条件にします。

参考資料

- RFC 9110、第9.2.2節：冪等なメソッド — HTTPの定義、冪等なメソッド、自動再試行の規則です。
- RFC 5789：HTTPのPATCHメソッド — PATCHが安全でも冪等でもない理由と、条件付き要求で安全に繰り返せるものにする方法です。
- Ansible：プレイブックの紹介 — Ansible自身の言葉による、望む状態、冪等性、チェックモードです。
- Ansible：commandモジュール — コマンドを安全に再実行できるものにする `creates` と `removes` です。
- PostgreSQL：INSERT — `ON CONFLICT` と、行がすでにある場合に `RETURNING` が返すものです。



用語集

実際の制約に合わせて選ぶ

代表的なデータセット、そこから答えを得たい問い、現在のボトルネックをお知らせください。品質、処理量、復旧の測り方を合意できます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「冪等とは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/idempotent/