日本ソフトウェア科学会第 29 回大会 (2012 年度) 講演論文集

Web API設計パターンの提案と既存Webサービスの分析

山崎 礼華　伊藤 恵　奥野 拓

Web サービスはネットワークを介して, 誰でも利用できるという点で有用である. しかし, その API(Web API) の設計が悪いと,Web サービスの中身が有用でも, 使い勝手が悪く, 使われない Web サービスになってしまう. 実際に Web サービスを構築する際に, どのように Web API を設計するべきか, その指針は明確ではない. 既存の API の分析を行い, 扱うデータの傾向ごとに違いが見られることが分かった. そこで Web サービスで扱うデータの傾向に着目し, 共通する項目を設計パターンとして定義した. そして提案した設計パターンがよく使用されている既存の API にどの程度現れているか, 調査を行い評価する. 本研究では扱うデータの性質に着目した Web API 設計パターンを提案し, これを利用することで経験豊富な Web API 設計者でなくとも使い勝手のよい Web API を設計可能にすることを目指す.

1　はじめに

Web サービスはネットワークを介して, 誰でも利用できるという点で有用である. 利用者が Web サービスを利用する場合には Web API を使用する. しかし, その API(Web API) の設計が悪いと,Web サービスの中身が有用でも, 利用者にとって使い勝手が悪いものだと思われてしまう. それゆえ使われない Web サービスになってしまう. では実際に Web サービスを構築する際に, どのように Web API を設計するべきかというと, その指針は必ずしも明確ではない.Web API は外部公開されて幅広い利用者に使われることが多いため, 外部設計が重視される. そのため内部設計と外部設計との, 両方の面から見たより良い設計が求められている. 人気の API/フレームワークを作るための 39 カ条 [1] では利用者にとって使いやすい API(フレームワーク, Web API を含む) のポイントがまとめられている. これを 1 つの指針として, 利用者にとって使いやすい Web API の設計を考えていく. そしてまとめた設計項目を設計パターンとして定義する. 複数の API を調査し, 提案した設計パターンに合致する API が, どの程度現れているか調査する. それにより提案した設計パターンの有用性を評価する.

2　関連研究

2.1　Web サービス

Web サービスとは, インターネット上で一般的には XML 形式等のメッセージを利用してシステムを連携させる技術のことである. Web サービスを利用することで様々なプラットフォーム上で動作する異なるソフトウェア同士が相互運用するための標準的な手段を提供することが可能である.

2.2　Web API

Web API とは, Web サイトなどの開発を効率的に行うための技術である.API は「application programming interface」の略であり, アプリケーションの開発者が, 他のハードウェアやソフトウェアの提供している機能を利用するための手法である. Web サ

A Proposal of Web API Design Pattern and Analyzing Existing Web Services with the Pattern

Reika Yamazaki, 公立はこだて未来大学大学院システム情報科学研究科, Graduate School of Systems Information Science, Future University Hakodate.

Kei Itou, Taku Okuno, 公立はこだて未来大学システム情報科学部, Faculty of Systems Information Science, Future University Hakodate.

イトなどの開発のためにインターネット経由で利用できる API のことを Web API という.

2.3 人気の API/フレームワークを作るための 39 カ条

API 設計のポイントが 39 カ条でまとめられている [1]. ここで述べられている API は広義な意味での API 設計についてまとめられているが, Web API も含まれているため, 1 つの指針として利用する. その 39ヶ条には人気の API にするための 26 カ条, API のドキュメントを充実させるための 7 カ条, 人気の API にする付加要素の 6 カ条に分かれている.

2.3.1 人気の API にするための 26 カ条

この 26 カ条では, 利用者からみた利用しやすい API とはどんなものであるのかポイントがまとめられている. また API の開発者が遵守すべきポイントもあげられている. 前半は主に利用者側にとって使いやすい API の要点が述べられている. 例えば, 「2. 頻繁に使うものは, デフォルト値で簡単に使えること」がある. Twitter API であれば最初から 20 件のツイートを取得できるように設定してある.20 件で問題ない場合はそのままデフォルトで使用することが可能である. また「5. 頻度の低い細かな機能や, 詳しい人は深いところまで細かく設定して使えること」という項目がある. 利用者に合わせた API の提供を考慮する必要がある. 後半は主に API 設計を行う開発者側にとっての要点が述べられている.

2.3.2 API のドキュメントを充実させるための 7 カ条

次に利用者が使いやすい API のドキュメントについて 7 カ条にまとめられている. 例えば「1. チュートリアルやサンプル, レシピが充実していると, 使い始めやすくなる」とあるように API の利用例を充実させる必要がある.

2.3.3 人気の API にする付加要素の 6 カ条

次には付加要素としてあるべき 6 カ条が記述されている. 多くの人が API を利用しやすくするために考えられる関連要素である. 例えば, 「3.API を利用する開発者間でコミュニティが形成されていること.(ノウハウを蓄積した人がいる, 利用者が多いといった安心感が得られる)」とあるように, API 独自の発展を考慮しての項目が見受けられる.

2.4 GoF のデザインパターン

デザインパターンとはプログラミング中に, 繰り返し現れるコードの記述に名前をつけ, 再利用しやすい形にまとめたものである [3] [4]. そうした設計の上でのルールをまとめることで, 誰しもが同じように使用することができる. 本論文では Web API 設計の上で必ず出てくるルールをまとめて, 誰でも同じように利用することで Web API の設計ができるものを目標とする. また, デザインパターンの形式についてもまとめられている.

パターン名と分類 パターンの本質を簡潔に表したパターン名を示す. パターンは目的 (生成, 構造, 振る舞い) と範囲 (クラス, オブジェクト) によって分類される.

目的 そのパターンが何をするためのものなのか, その意図を記述する.

別名 よく知られた別名があれば, それを述べる.

動機 そのパターンが, どのような問題を解くのかのシナリオを記述する.

適用可能性 そのパターンが, どのような状況で利用できるかを記述する.

構造 そのパターンのクラスを, 図形的な表記法 (ダイアグラム) で表記する.

構成要素 そのパターンを構成するクラスやオブジェクトと, それらの責任分担を述べる.

協調関係 それぞれの構成要素間の関係性を示す.

結果 パターンを利用した結果, どのようになるかを記述する.

実装 パターンを実装する際に注意するべき事項を述べる.

サンプルコード パターンをどのように実装するかをコードで例示する.

使用例 実際のシステムでつかわれているパターンの例を,2 つ以上示す.

関連するパターン そのパターンと関連したほかのパターンについて示す.

以上がパターンにかかせない点として挙げられている. これらの条件を含むデザインパターンが望まれる. しかし, GoF のデザインパターンは実際にソフトウェアを内部設計する段階に至ってのものが多いため, 本研究ではデザインパターン自体は利用しないが, パターンのまとめ方の 1 つの指針として利用する.

3 設計パターンの提案

Web API の設計パターンの候補となるものを見つけ出すため, 既存の Web API の分析を行った. 分析にあたって API が扱うデータの傾向に着目した. 本稿では更新頻度の高く, データ量の多いものを扱う API に関する設計パターンについて述べる.

3.1 データの傾向と望ましい API

API は内部的な ID を極力使わないようにして,ID を知らないとユーザが利用できないことをなくすことが望ましい. それにより ID を知らずとも API を利用できるようになり, ユーザにとって利用しやすいものになるだろう. またデータの取得の仕方では, 更新頻度が高いと, 過去のデータを取り出すよりも, 最新のデータを簡単に取り出せる方が利用するユーザにとって便利である. そのため最新のデータを取り出す API は必要とされ, デフォルトで取り出す件数が指定されていると, 妥当数を取り出すことが可能である. さらに, 複数データを取り出す場合であれば, データの一覧を取得し, 個々のデータにアクセスできるようにしてあると, 大量のデータの処理に困ることを防ぐ.

3.2 設計パターンの定義

必要となる Web API の設計パターンを, 以下の項目に沿って定義する.

パターンの名前 パターンの本質を簡潔に表したパターン名を示す

概要 どういう状況で利用するか示す

API 例 どういう API を構成すべきか示す

例として 1 つ 1 つのデータサイズは大きくないが, 更新頻度が高く, データ量の多いデータを利用する場合に望ましい Web API パターンを提案する.(表 1)

表 1 の API 例のうち,「最新のデータを取得する API」とはデータのうちの最新のデータを 1 つ, またはいくつかを取得する API である. パラメータ無しでも妥当と思われる個数の最新のデータを取得できる.「過去のデータを取得する API」とはデータのうちの過去のある範囲のデータを 1 つ, またはいくつかを取得する API である. 範囲の指定にはデータに含まれる日付等が利用される. 「データの検索をする API」とはある検索条件を設定し, その条件に当てはまるデータを取得する API である. この 3 つの API が必要になるだろう. この 3 つに対応する API があれば, サービスを十分に活用できるであろう.

4 既存 API の分析による設計パターン評価

既存の Web API に提案する設計パターンに合致する API が, どのくらい現れているのか調査することで, 設計パターンとしての有効性評価を行う. ある設計パターン i の適用率を以下の式のように定義する. そのため, まずは前提条件を満たす API 数を調査する.

$$\text{設計パターン}_i\text{の適用率} = \frac{\text{設計パターン}_i\text{に合致する } API \text{ 数}}{\text{設計パターン}_i\text{の前提条件を満たす } API \text{ 数}}$$

4.1 API の調査

表 1 で述べた設計パターンの前提条件を満たす API に, どのようなものがあるかを,WAFL [2] という Web API 紹介サイトに紹介されている Web API を対象として調査を行う.WAFL には 200 を超える API が紹介されており, API 情報が集約されている. WAFL に登録されている API のうち, 現在もサービス提供されている 182 個を対象とし, 更新頻度が高くデータ量の多いサービスに該当するものはいくつあるか調査した.(表 2)

更新頻度が高くデータ量の多い API に該当するものは 182 個中 65 個であった. 該当するサービスの種類には Web ページを対象としたものや, ニュースサイトなどがあった. 次にサービスを利用しているユーザ数や登録されている口コミ数により, 更新頻度が変化するものは 182 個中 20 個であった. これはお気に入りユーザの情報を取得するため, お気に入りユーザ

表 1　提案する設計パターン (一部)

パターンの名前	更新頻度の高いデータへのアクセス
概要	個々のデータサイズは大きくないが, 更新頻度が高く, 量が多いデータにアクセスする際に使用する. 個々のデータにはそのデータの作成日時等の日時情報が含まれることを前提とする. 個々のデータが内部的な ID を持っていても良いが, サービス利用者が内部的な ID をできるだけ使わなくて良いように API を設計する必要がある.
API 例	以下の 3 つの API で構成する. 1. 最新のデータを取得する API API 名　RecentData 概要　そのサービスで扱っているデータのうち, 最も新しいものから一定個数分を取得する. パラメータ　取得する個数か, 取得する最初のデータを指定するための日時等の条件を指定するパラメータを用意する. ただし, このパラメータは必須とせず, パラメータが指定されていなくても, そのサービスにとって妥当と思われる個数の最新のデータを取得できるようにする. レスポンス　データサイズが比較的小さい場合はデータ列をそのまま返す. データを直接返すのが適切でない場合は, 個々のデータを特定する ID リストを返すこととし, 別途 ID からデータそのものを取得する API を提供する. 2. 過去のデータを取得する API API 名　PastData 概要　そのサービスで扱っているデータのうち, 始点と終点で示される範囲のデータ列を返す. パラメータ　そのサービスにとって妥当な始点と終点を指定する必須パラメータを用意する. 例えば, 最初と最後の日時を指定してその範囲内のデータを取得する. レスポンス　データサイズが比較的小さい場合はデータ列をそのまま返す. データを直接返すのが適切でない場合は, 個々のデータを特定する ID リストを返すこととし, 別途 ID からデータそのものを取得する API を提供する. 3. データの検索をする API API 名　DataSearch 概要　そのサービスで扱っているデータに対して, パラメータで指定する条件に合致したものだけをデータ列として返す. パラメータ　そのサービスで扱っているデータに対して適当な条件を必須パラメータとして指定する. 例えば, キーワードを指定してキーワード検索を行うなどが考えられる. レスポンス　データサイズが比較的小さい場合はデータ列をそのまま返す. データを直接返すのが適切でない場合は, 個々のデータを特定する ID リストを返すこととし, 別途 ID からデータそのものを取得する API を提供する.

表 2　更新頻度が高くデータ量の多い API 数

分類	個数
該当する	65
ユーザ数や口コミ数などに依存	20
該当しない	90
不明	7

の数に依存するものが挙げられた. また該当しないものは 182 個中 90 個であった. 該当しないものには計算式で情報を返したり, 緯度経度を渡すことで地図を表示するものが挙げられた. そのほか不明なものとしては 182 個中 7 個あり, 現在データの更新を停止しているため更新頻度の調査が行えないものがあった.

4.2　ニュースサイトの API

調査した API のうち更新頻度が高くデータ量の多いもののうち, 提案する設計パターンに合致するか調査している. そのうちの 1 つとしてあるニュースサイトの API の分析結果を述べる.

4.2.1　既存 API の構成

このニュースサイトにはトピックス API, トピックス見出し API, トピックスアーカイブ API の 3 つの API がある. トピックス API は検索キーワードを指定し, 該当する最新のデータを返す API である. トピックス見出し API はトピックの掲載された期間を指定し, 期間内に掲載されていた見出しに関してキーワードで検索する API である, また, レスポンスをアクセス数が多かった順などでソートすることもでき, さらには, それぞれの見出しが掲載開始から終了までの間にどのように注目されたのか, 時系列で見ることもできる. トピックスアーカイブ API はカテゴリよりも細かい分類基準であるトピックに関するデータを扱っており, 期間を指定することでその期間内のトピックに関してキーワードで絞り込む API である. また, レスポンスをアクセス数が多かった順などでソートすることもできる. さらには, それぞれのトピックが期間内でどのように注目されたのか, 時系列で見ることもできる. 例としてトピックス API を取り上げ, 詳細を表 3 にまとめた.

トピックス API では検索キーワードに一致または部分一致するものを, 見出し, トピック名, トピック概要, キーワード, サブジャンルから検索する. そして検索結果を最大 10 件返す. またカテゴリを指定することで検索条件を絞ったり, 検索結果をソートして返すこともできる. 通常は最終更新時間がソート順になっており, 最新のニュースが 10 件取得される. しかし, 通常の検索結果は 10 件しか返さないが, 先頭位置を指定することで 11 番目から 20 番目の検索結果を取得することもできる. また, トピックス API のレスポンスについて表 4 に詳細を記載する. トピックス API のレスポンスではトピックに関する様々な情報が記載され, 個別レスポンスは取得する件数分表示される.

4.2.2　設計パターン合致性調査

このニュースサイトの API の内部と提案した設計パターンとを比較する. トピックス API は通常, 最終更新日によってソートされるため, 最新のデータが返される. これは提案した設計パターンの「最新のデータを取得する API」にあたる. 提案したパターンでは最新のデータを 1 つ, またはいくつかを取得する API である. パラメータ無しでも妥当と思われる個数の最新のデータを取得できるとした. トピックス API では通常では最大 10 件が取得できるとあり, サービスにとって妥当な数を指定していると思われる. また, 取得するデータの先頭位置を変えることで, 最新のデータではなく過去のデータを取得することができる. これは「過去のデータを取得する API」にあたる. さらには, 該当するニュースに関してのデータを取得するために, キーワードによる検索が行えるようになっている, これは.「データの検索をする API」にあたる. トピックス API では他にトピック名を指定したり, カテゴリを指定して検索を行うことができる. 提案した設計パターンに当てはまるデータの扱いができるため, トピックス API は設計パターンを満たしているといえる.

5　今後の展開

現在一部の設計パターンを提案しているが, 今後はより多くの設計パターンを提案し, 様々な種類の Web API に対応できることが望まれる. そのためには調査

表 3 トピックス API リクエストパラメータ

パラメータ	値	説明
appid(必須)	string	アプリケーション ID
topicname (いずれか必須)	string	トピック (国内や経済などのカテゴリよりも細かい分類基準) 名の絞り込み指定. 英字表記で,URL の末尾部分と対応
category (いずれか必須)	string	カテゴリ (国内や経済など大まかな分類基準) の絞り込み指定 英字表記で,domestic(国内),world(海外) 等. 指定がない場合はすべてのカテゴリ
pickupcategory (いずれか必須)	string	掲載されたカテゴリを指定 (上の category におけるカテゴリと一致しない場合あり). 英字表記で,domestic(国内),world(海外) 等. 指定がない場合はすべてのカテゴリ
query (いずれか必須)	string	UTF-8 で URL エンコードされたワードで該当するトピックを検索 見出し, トピック名, トピック概要, キーワード, サブジャンルが検索対象で, 部分一致
relatedinformation (いずれか必須)	integer	各トピックの関連情報を取得するかどうかを指定. 指定がない場合は取得しない. 0 ：取得しない (デフォルト) 1 ：目次情報取得 2 ：全文取得 (topicname の指定必要)
sort	string	レスポンスの表示順を指定. 以下のいずれか. 指定がない場合は最終更新時間が新しいものから順に表示 pvindex ：PV 指標順 pickup ：掲載時の表示位置順 (pickupcategory とあわせて指定) datetime ：最終更新時間順 (デフォルト) relatedinfotime ：関連情報更新順 headlinestime ：ヘッドライン更新順 newsnum ：関連ニュース件数順
results	integer	表示件数の指定 最大値は 10 件で, 指定がない場合は 10 件
start	integer	結果の先頭位置を指定 11 件以上の該当データがある場合などに指定 指定がない場合は 1 件目 (最初) から

する API を増やすことが必要になる. そして調査した API と設計パターンとの評価を行う. また, 提案したパターンの有用性についても評価も行っていく. 評価方法は, 2 通り考えている. 1 つは既存の API に対し, 提案する設計パターンの通りに設計されているのか比較する方法である. もう 1 つは提案する設計パターンを適用して Web API を設計する. そして完成した Web API を評価することで, 設計パターンが有用かどうかを評価する. いずれか, または両方の方法を用いて, 設計パターンの評価を行う.

表 4　トピックス API レスポンスパラメータ

フィールド	説明
ResultSet	クエリーレスポンスのすべて totalResultsAvailable ： データ内のマッチしたクエリー数 totalResultsReturned ： 返却され, かつマッチしたクエリーの数 firstResultPosition ： 全検索結果の最初のポジション
Result	個別レスポンス
HeadlineId	トピックの見出し (Title) に対応する ID
DateTime	最終更新日時 (ニュース, ヘッドライン, 関連情報のいずれかで一番最新の更新があった日時)
CreateTime	トピック (国内や経済などのカテゴリよりも細かい分類基準) が作成された日時
NewsUpdateTime	トピックのニュースの最終更新日時
RelatedInfoUpdateTime	トピックの関連情報の最終更新日時
HeadlineUpdateTime	トピックの見出し (15 文字程度のテキスト) に対応して更新されるヘッドラインの最終更新日時
Title	トピックの見出しない場合は表示されない
Keyword	トピックに関連するキーワード 最大 5 件まで
Word	具体的なキーワード
TopicName	トピックの日本語表記随時更新される見出しとは異なり, 基本的に固定の名称
English	トピックの英語表記これは固定の名称
Overview	話題の単位であるトピックについての数十文字の簡単な説明
Category	トピックが所属するカテゴリ (国内, 海外, 経済等のいずれか)
SubCategory	サブカテゴリ (社会, 政治などカテゴリの下の分類指標). 複数の場合あり
Sub	具体的なサブカテゴリ名
Url	トピックの URL
PickupCategory	掲載されたカテゴリ (上の Category におけるカテゴリと一致しない場合あり)
PickupOrder	掲載されたときの表示順位 (「主なトピックス」掲載時の順位のみ)
PvIndex	PV 指標は現在のアクセス数から割り出した指標的な数値
EditNum	関連情報がエディターによって更新された回数
NewsNum	掲載されているニュースの件数
NewsUrl	ニュース一覧ページの URL
RelatedInformation	関連情報 (最大 10 件まで)
TotalNum	関連情報の帯 (関連情報エリア内の大見出し) の数
RelatedInfoTitle	関連情報の帯名
RelatedInfoUrl	関連情報の帯別の URL
RelatedInfoText	関連情報の帯ごとの内容 (本文) Wiki 文法 (関連情報独自の簡易な記述言語) 使用
SmartphoneUrl	スマートフォン最適化ページの URL

6 まとめ

Web サービスが扱うデータの傾向に着目し, 設計パターンを提案した. そして既存 API のうち更新頻度が高く, データ量の多いものを扱う API 数を調査した. そして調査した API が提案した設計パターンをどの程度満たしているか評価を行った. 今後も調査を続け, 設計パターンの適用率を調査する.

参 考 文 献

[1] 安藤幸央, 人気の API/フレームワークを作るための 39 カ条, http://www.atmarkit.co.jp/fjava/column/andoh/andoh35.html, 2007/7/20
[2] VISH 株式会社 名古屋本社, ワッフル/WAFL, http://wafl.net/, 2012
[3] 江渡浩一郎, パターン,Wiki,XP 時を超えた創造の原則, 株式会社技術評論社, 2009
[4] Erich Gamma, Ralph Johnson, Richard Helm 他, オブジェクト指向における再利用のためのデザインパターン, ソフトバンククリエイティブ, 1999