青山学院大学理工学部情報テクノロジー学科
2016 (平成 28) 年度 卒業論文要旨

REST に基づいたリアルタイムアプリケーションフレームワークの設計

矢舗 宗一郎 (15813088)
Dürst 研究室

1 研究背景

Roy Thomas Fielding によって提唱された REST(Representational State Transfer) という Web アーキテクチャスタイルが存在する. Web は REST に沿って設計することで, サービス全体を効率的に, 可用性・信頼性高く開発することができる.

近年, Twitter や LINE など, 動的にリソースを更新するアプリケーションが増加している. それに伴い, クライアントとサーバ間の通信速度の高速化やリソース更新の最適化に対する需要が高まっている. これらを実現するリアルタイム通信技術として, WebSocket や Server sent Event などが挙げられる.

しかしこれらの技術には, REST で必要とされているアドレス可能性, 接続性, 統一インターフェス, ステートレス性が欠如している. 本研究では, リアルタイム通信アプリケーションの Web との親和性改善を向上させるため, リアルタイム通信を利用することで発生する問題を分析し, 新たなアーキテクチャスタイルを提案する.

2 基礎技術

本章では, REST, リアルタイム通信技術, 実装に用いたライブラリやフレームワークについて述べる. リアルタイム通信技術については, 実装に用いた WebSocket を中心に述べる.

2.1 REST

REST とは, Representational State Transfer の略称である.

REST は, Roy Fielding が Web の成功要因についてアーキテクチャスタイルとしてまとめたものである. Web のアーキテクチャスタイルには, 他にも SOAP などいくつか存在するがここで紹介している REST が現在の主流となっている. Web サービスや Web API が REST の規約を守ることで Web 全体の調和が取れる. Roy Fielding が論文の中で提案した REST は, HTTP などに技術を限定しておらず幾つかのアーキテクチャから構成されている. 本章では, リソース, REST を構成するアーキテクチャ, RESTful なアプリケーションの性質, 利点を述べる.[1]

2.1.1 アーキテクチャ

クライアントサーバシステムクライアントとサーバの二つの役割に分割する分散型のコンピュータシステムである. サーバはアプリケーションのリソースを管理し, サービスを提供する役割を果たす. それに対し, クライアントはユーザとのインンターフェース部分を担当する. このような構成にすることによって, アプリケーションの大部分をサーバに任せることによって, クライアント側ではサービス管理が容易となる.

2.1.2 ステートレス

サーバがクライアントのセッション状態を持たないという制約である. クライアントのセッション管理しないことで, サーバの負荷を軽減し, クライアントのリクエストに対し常にリソース操作に対する必要十分な情報のレスポンスを渡すことができる. またクライアントの状態に依らないので処理がシンプルとなる. これらによってスケーラビリティの向上を望むことができる.

2.1.3 統一インターフェース

リソース操作を統一された限定的なインターフェースで行うアーキテクチャのこと. これによってサーバやクライアントが変わっても, 常に同じインターフェースが使われることが保証される. また全体のアーキテクチャがシンプルになり, 可読性も増すため, REST にとって中心的な制約であると Roy Fielding は位置付けている.

2.1.4 キャッシュ

キャッシュとは, 一度取得したリソースをクライアントが再利用する仕組みである. これを利用することで, サーバとクライアント間の通信を削減しネットワーク帯域を節約することができる. またデータの再取得の必要がないため処理を高速化可能である. 頻繁に利用するリソースであればあるほど恩恵を受けることができる. サーバサイドのキャッシュ機能は, 本研究でフレームワークとして使用している Ruby on Rails でも実装されている.

2.1.5 階層化システム

システムをいくつかの階層に分割するアーキテクチャのこと．階層化によって設計がシンプルになりスケーラビリティを向上する．各システムで他のシステムをほとんど無視しながら開発を進めることができる．このようなことを実現することができるのは, インターフェースが統一されているからである．

2.1.6 コードオンデマンド

クライアントがサーバからプログラムコードを受け取り, それをクライアント上で実行するアーキテクチャスタイルのこと．JavaScript や Flash がこれにあたる．これによってクライアント側のみで処理することが増える．これによってサーバはストレージとしての処理に集中することができる．またユーザに対する処理速度が向上する．

2.2 リソース

リソースとは Web 上に存在するデータのことを指す．またそれ自体を参照するに値するほどの重要性を持っているもの．リソースに一意な名前を与えることによって, リソースを識別することができる．この一位な名前に使われているフォーマットが URI である．

2.3 RESTful なアプリケーションの性質

REST の規約に従った設計のアプリケーションを RESTful なアプリケーションと呼ぶ．RESTful な Web アプリケーションの性質として, アドレス可能性, 接続性, 統一インターフェース, ステートレスがある．これらの性質の重要度は上記の順番で高い．この理由として, 接続性はアドレス可能性が保証されていることで実現することができること．統一インターフェースはアドレス可能性と接続性が成り立つことで効果が発揮されることがあげられる．また多くのアプリケーションでユーザのセッション情報を使用する必要があるため, ステートレスを厳密に実現することは難しい．このためステートレスの重要度は下がっている．

2.3.1 アドレス可能性

URI を使用して, リソースを簡単に指し示すことができる性質のこと．リソースには名前がついている方がプログラムの処理がしやすくなる．また URI を付加することによってリソースに接続性をもたらすことができる．そのためリソースには URI を付加することが大事なのである．ただしリソースとして情報が多すぎるという場合, それらに対して粒度を決めて表現することが重要隣ってくる．

2.3.2 接続性

リソース間で遷移可能な性質である．REST の基幹をなす思想で, リソースをリンクで接続することで 1 つのアプリケーションを構成するという考え方である．REST では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 (Hypermedia as the engine of application state) という。つまり、ハイパーメディアのリンクをたどる作業が REST に影響を与えているのである。

2.3.3 統一インターフェース

通常 Web では, HTTP が使用される．その中では, GET, POST, PUT, DELETE の 4 つのメソッドが主に使用されている．このように使用するメソッドは限定するべきであると考えるのが統一インターフェースの性質である．またこれらのメソッドは限定するだけでなく, 正しく使用するということが重要である．これらによってアプリケーションの仕様が統一され, 設計のコストや可読性も増す．

2.3.4 ステートレス

サーバでクライアントのセッション状態を保持しないという性質である．これによって, HTTP リクエストが分離しているとサーバが複数存在する場合にも, どのサーバにアクセスしても同じ結果となることが保証され, スケーラビリティが向上する．またサーバが状態を管理していないため, 実装も簡潔になるという利点が存在する．近年では, cookies の使用などによるセッション管理も行われている．RESTful では, 完全なステートレスではないが cookies によるセッション管理は認めている．

2.4 WebSocket

WebSocket とは TCP コネクションを使用した双方向通信プロトコルである．そもそもは HTML5 の一部として策定していたものであったが単体プロトコルとして仕様の策定を推進したもの．従来の HTTP 通信 と比べ, 通信コストが低い点やリアルタイム性が高い点が評価されている．[2]

2.4.1 コネクション確立

コネクションを確立する際には, クライアントから要求を送りサーバから応答を受けることで可能となる．その際の要求と応答は HTTP の際の仕様と変わらない．詳しくは, クライアントの要求における Upgrade ヘッダフィールドの値を WebSocket とすることで実現を可能にする．[3]

2.4.2 用意されているイベント

以下の 4 つは WebSocket オブジェクトで定義されているイベントである．

メッセージを送信する際には send メソッドを使用する．

表 2.1: 用意されているイベント

コールバック	イベント
onopen	接続を開始した時のイベント
onmessage	メッセージが送られた時のイベント
onerror	エラーが発生した時のイベント
onclose	接続を閉じた時のイベント

2.5 実行環境

2.5.1 Rubu on Rails

Ruby on Rails は, Ruby を使用した Web アプリケーションフレームワークである. このフレームワークには大きく二つの特徴がある. DRY と CoC である. DRY は, Don't repeat yourself の略称で, 同じ記述を繰り返さないという制約である. これによって重複の削減や, システムの複雑化などを防ぐことができる. また CoC は, Convention over configuration の略称で, 設定より規約をという制約である. 規約を守り設計することで, 生産性の向上を望むことができる.

2.5.2 JavaScript

Web 上でインタラクティブな表現をするために開発されたオブジェクト志向のスクリプト言語である. 従来の Web ページは, 静的な表現しかできなかったのに対し, JavaScript の登場により幅広い表現が可能となった. HTML 内にプログラムを埋め込むことができて, Web ページに様々な機能を付加することができる.

2.5.3 ajax

Asynchronous JavaScript + XML の略称. JavaScript の組み込みクラスである XMLHttpRequest を利用した非同期通信を利用して, ウェブブラウザ上で非同期的なインターフェースを実現することを総称している.

2.5.4 websocket-rails

Rails 上で動作する WebSocket ライブラリである. イベントと呼ばれる単位で, サーバとクライアント間の送受信を行う. サーバサイドでは Event Router という仕組みで, イベントを対象のアクションにマッピングし, クライアントサイドは WebSocket-Rails 専用の API を使用してイベントのやりとりを行う.

3 提案手法

本研究では, 複数のリアルタイムアプリケーションに REST の性質を追加する. リアルタイムアプリケーションは非リソース志向でステートフルな通信となる.[4] そこに REST の性質を追加する. 具体的にはアドレス可能性, 接続性, 統一インターフェース, ステートレス について検討する. 次にこれらの共通項を抽出した後, それらをフレームワークとして実装する.

4 実装

作成予定のリアルタイムアプリケーションは, 複数人でプレイ可能な壁で反射するボールゲームと Google ドキュメントのような複数人で同時に編集可能なアプリケーションである. 本研究の前任者である繪面がリソースに対するアドレスの付加方法やインフラ構成において様々な提案, またその見解を述べていた [5] のでそれに関する検討, 考察を行いながら進める.

現段階では, REST の性質を加えたチャットアプリを作成した. チャットアプリのインフラ構成は図 1 のようである.

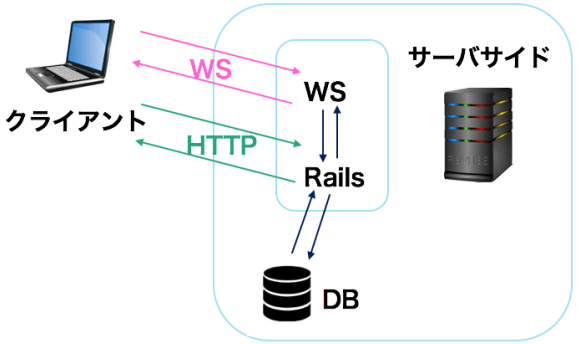

図 1: インフラ構成

開発環境は, サーバサイドは Ruby on Rails, websocket-rails, クライアントサイドは Ruby on Rails, JavaScript, websocket-rails で実装している.

チャットアプリはメッセージを保持するテーブルを用意しており, リソース取得のリクエストが来たらそれに相当する Web ページを HTTP で返す. メッセージを保持しているテーブルは表 4.2 の通りである.

表 4.2: メッセージテーブルのカラム

カラム
id
body
user_id
created_at
updated_at

5 評価

チャットアプリを開発する上でステートレスの性質を加えることに関して疑問が残った. 最近では Cookie によるセッション管理を使用した Web サービス や Web アプリも多く存在する. Rails4 以降では, デフォルトでセッション情報がブラウザの Cookie に保存される.

またアドレス付加の方法, データの保持方法, リソースの表示方法も検討中である. 今回作成したチャットアプリでは, リソース一つ一つに id を振り参照するというものだった. データはメッセージと投稿者情報を保持していてそれをリソースとして表示した. リソースにはある程度粒度を持って URI を付加するべきかと考えている. 残り二つの実装をしてみてからデータの保持方法とリソースの表示方法について検討する.

6 今後の予定

複数人でプレイ可能な壁で反射するボールゲームの実装を行う. 10 月の下旬までに前回の開発で得た疑問点や検討すべきポイントを考慮しながら完成させる. 11 月までに Google ドキュメントのような複数人で同時に編集可能なアプリケーションを完成させる.

参考文献

[1] Leonard Richardson and Sam Ruby. RESTful Web サービス. オライリー・ジャパン, 2007.

[2] Lok Fang Fang Stella, Stan Jarzabek, and Bimlesh Wadhwa. A comparative study of maintainability of web applications on j2ee,. net and ruby on rails. In *Web Site Evolution, 2008. WSE 2008. 10th International Symposium on*, pp. 93–99. IEEE, 2008.

[3] Ian Fette. The websocket protocol. 2011.

[4] Alex MacCaw. ステートフル JavaScript - MVC アーキテクチャに基づく Web アプリケーションの状態管理. オライリー・ジャパン, 2012.

[5] 繪面友香. Rest とリアルタイムアプリケーションの統合フレームワークの設計. 修士論文, 青山学院大学, 2015.