マッシュアップ型リッチクライアントサービス開発モデルの提案

横井　公紀†　　中村　浩二†　　中道　上†　　青山　幹雄†
南山大学　数理情報学部　情報通信学科†

1. はじめに

近年 Web API の公開が増加し，複数の Web API を組み合わせ，手軽にサービスを開発するマッシュアップが注目を集めている[3]．本稿では，マッシュアップに焦点を当て，Web API を用いたリッチクライアントサービスの実現を目的とした開発モデルを提案する．

2. マッシュアップ型開発の問題点

MVC モデルの観点から，マッシュアップ型開発は View の重ね合わせとなるため，Web API の処理の実行制御ができない．そのため，より高度なマッシュアップ開発のためには Model の制御を行う仕組みが必要である．

3. 関連研究

(1) Web サービス連携

Web サービス連携では，実行前にプロバイダによって組み合わせるサービスと順序が決定されている．

(2) ツールによるマッシュアップ開発支援

マッシュアップを支援するツールが提案されている．例えば MashMaker[2]では，Widget が複数の Web API のレスポンスを View に反映する．しかし複数のサービスの実行制御は行えず，View の重ね合わせに限定される．

4. アプローチ

マッシュアップ型開発を MVC によりモデル化し，重ね合わせの対象と方法を提案する．

4.1. 前提条件

(1) 複数の Web API の連携を前提とする．
(2) Web API へのリクエストは REST で行う．

4.2. MVC に基づいた重ね合わせ対象の分析

(1) Model の重ね合わせ

複数の Web API が提供する機能を統合する．しかし，Model の重ね合わせは困難である．

(2) Control の重ね合わせ

制御順序をロジックに記述することにより，Model と View の両者を制御する．

(3) View の重ね合わせ

従来のマッシュアップ型開発と同様，Web ページ上でデータを重ね合わせる．

以上の議論より，View および複数の Model の制御が可能と考えられる Control の重ね合わせが有効である．

4.3. 重ね合わせ方法の決定

Control の重ね合わせには，クライアントあるいはサーバでマッシュアップを行う方法がある(図1)．クライアントサイドの場合，リクエスタにモジュールを組み込む方法がある．サーバサイドの場合，リクエスタと Web API の間でブローカが双方の情報を仲介する方法がある．

(1) リクエスタ内への制御プログラム組込み

リクエスタはモジュールの使用をサポートしている必要がある．また，処理量が増加すれば多くのリソースが必要となる．

(2) ブローカへの制御プログラム組込み

ブローカに制御順序を記述したプログラムを組み込む．リクエスタ側に変更を加えることなく利用できる．

このため，本稿では(2)の Control の特徴を考慮し，ブローカを用い制御を仲介するアーキテクチャを提案する．

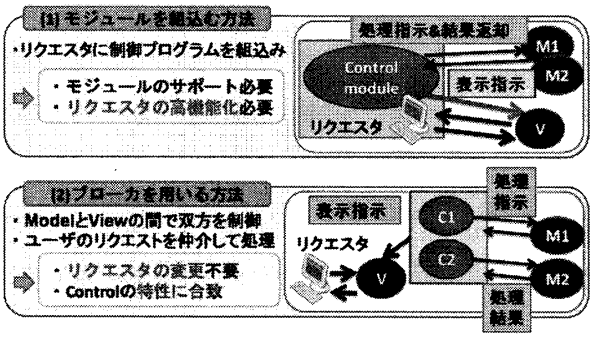

図 1　Control の重ね合わせ方法

5. Control ブローカの提案

5.1. ブローカのアーキテクチャ

ブローカのアーキテクチャを図 2 に示す．

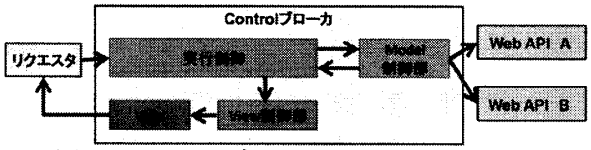

図 2　Control ブローカのアーキテクチャ

ブローカでは実行制御，Model の制御，View の制御を行う．実行順序は，Model と View の制御をロジックによって重ね合わせる．また，リクエスタからのリクエスト情報に基づき，Web API へリクエストを行う．各制御部の機能の定義を以下に示す．

(1) 実行制御

制御順序に従って各制御部を呼び出す．

(2) View 制御部

実行制御で得られた結果を View に表示する．

(3) Model 制御部

Web API へのリクエストとレスポンスの処理を行う．

A Development Model of Rich Client Services Based on Mash-up
†Kiminori Yokoi, Koji Nakamura, Noboru Nakamichi, Mikio Aoyama, Nanzan University.

5.2. Web API の仕様に関する考慮点

ブローカの実現には Web API のリクエスト形式などの仕様の差異と，リクエスト URL などの固有の情報に関する問題がある(図 3)．これらの問題を解決するために，仕様差異部分を吸収したクラスを開発し，固有の部分を記述する．

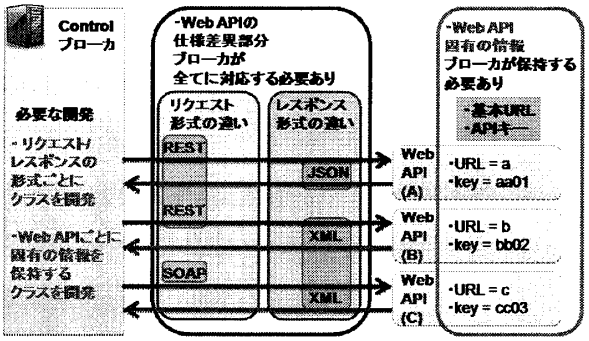

図 3　Web API の仕様に関する考慮点

6. プロトタイプの開発

6.1. プロトタイプのアーキテクチャ

プロトタイプにより提案手法の妥当性を評価する．プロトタイプのアーキテクチャを図 4 に示す．

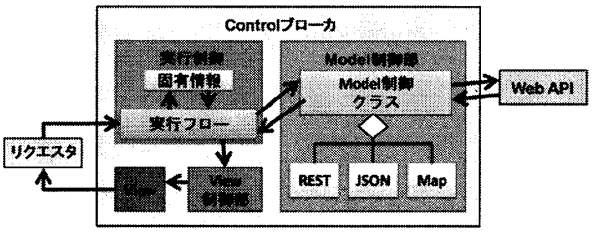

図 4　プロトタイプのアーキテクチャ

プロトタイプにはリクルート社が提供するグルメ情報サービス"HotPepper"の Web サービス(http://webservice.recruit.co.jp/)のグルメサーチ Web API を利用した．

実行制御はロジックを記述した実行フローを持ち，各制御部を通じて複数の Web API を制御する．まず，リクエスタから送られたキーワードを基に，Web API の固有情報からリクエスト URL を生成し，Model 制御部へ送る．Model 制御部は Web API の仕様を変換する 3 種類のクラスでレスポンスを生成し，実行制御に返す．その結果を View 制御部が表示する．

実装は，アプリケーションサーバ環境に Apache Tomcat (6.0.18)を採用し，Java Servlet(JDK 1.6.0, Servlet 2.5, JSP 2.1)を用いて開発した．7 個のクラスを実装し，規模は 418 行となった．

6.2. プロトタイプの実行例

開発したプロトタイプの実行例を図 5 に示す．提案手法に基づいたプロトタイプにより，ユーザのリクエストを基に Web API から処理結果を受け取り View に表示するまでのシナリオの実行が確認できた．

また，図 5 の実行例は実行フローの記述に基づいている．実行フローの記述を変更することにより，Web API の制御を変更することが可能となった．

図 5　プロトタイプ実行例

7. 評価と考察

(1) クライアントサイドマッシュアップをサーバに移行

提案手法はサーバサイドでマッシュアップを行うことで，JavaScript による開発において問題であったコードの秘匿性を確保した．また Java Servlet による実装により，ブラウザの種類に非依存な動作を実現した．

(2) Web API の Control の重ね合わせを実現

複数の Web API の Control を重ね合わせるブローカアーキテクチャを提案した．プロトタイプの実装を行い，制御の変更が可能であることを確認した．

(3) Web API のインタフェースに対する再利用性の確保

Web API の仕様差異部分と固有部分を明確に分離した．仕様差異部分を記述したクラスを集約した新しいクラスに固有の部分に関する記述を行い，処理の正常性を確認した．さらに，集約したクラスは同一の Web API を用いた開発であれば再利用可能である．

(4) ロジックを持つマッシュアップ開発の簡易化

固有部分を分離したことにより，実行フローの記述はこれらの手続き呼び出しで実現できる．固有部分を記述したクラスを揃えることにより，実行フローの複雑なビジネスロジックの記述が不要となった．

8. 今後の課題とまとめ

本稿では MVC モデルに基づいた分析から，ブローカによる Web API の Control の重ね合わせを提案した．

今後，View の制御に関するアーキテクチャの詳細化を行うことにより，リッチクライアントサービス実現への応用が期待される．

参考文献

[1] 中村 浩二，横井 公紀，マッシュアップ型アプリケーション開発モデルの提案, 南山大学 2008 年度卒業論文, 2009.

[2] R. Ennals, et al., MashMaker: Mashups for the Masses, Proc. 2007 ACM SIGMOD, Jun. 2007, pp. 1116-1118.

[3] X. Liu, et al., Towards Service Composition Based on Mashup, Proc. 2007 IEEE Congress on Services, Jul. 2007, pp. 332-337.