6X-1

クライアントで動作するWebAPIの実装と評価

竹渕 瑛一† 山田 泰宏† 鈴木 浩† 服部 哲† 速水 治夫†
†神奈川工科大学大学院情報工学専攻

1 はじめに

WebAPIとは，Webアプリケーションを Webから利用するための APIである．昨今，いくつかの WebAPIを組み合わせることで，新たな Webアプリケーションをマッシュアップする事例が増えている．

WebAPIは，クライアントから送られたリクエストを受け取ると，Webアプリケーションと連携してリクエストに応じた処理を行い，処理結果（レスポンス）を JSONや XMLなどの形式でクライアントへ返す．

WebAPIはサーバで動作する．これはレスポンスを生成する過程でシステムの内部処理をブラックボックス化するためである．最近，WebAPIが提供している機能の一部を制限する事例がある．例えば，TwitterAPIでは，サーバの負荷を軽減するため，従来から使用されてきた機能の一部を削除している [1]．WebAPIの機能はサーバに多大な負荷がかからない限り，高い自由度を持っていることが望ましい．

そこで，Webアプリケーションの HTML文書を取得するためのクライアントライブラリを利用する手法を提案する．一般的にクライアントライブラリは WebAPIの機能を特定のプログラミング言語から利用するためのライブラリである．提案手法のクライアントライブラリは Webアプリケーションの HTML文書を取得する．

提案手法は WebAPIの課題であった自由度とサーバの負荷軽減の問題を両立するものである．提案手法は取得した HTML文書をクライアントライブラリによって加工するため，必要に応じてライブラリ利用者が拡張，改善することができる．また，HTML文書のみを取得するため，レイアウトに使われる画像や広告などに対してリクエストが送られなくなるため，Webブラウザで Webアプリケーションにアクセスする場合と比べるとサーバの負荷が大幅に軽減される．

Implementation and Evaluation of WebAPI operate on the client
Eiichi Takebuchi†, Yasuhiro Yamada†, Hiroshi Suzuki†, Akira Hattori†and Haruo Hayami†
†Graduate School of Information Engineering, Kanagawa Institute of Technology

2 提案手法

本章では提案手法の構成や実装方法について述べる．

2.1 提案手法の構成

提案手法のクライアントライブラリはクライアントのアプリケーションにインポートすることで動作する（図 1）．

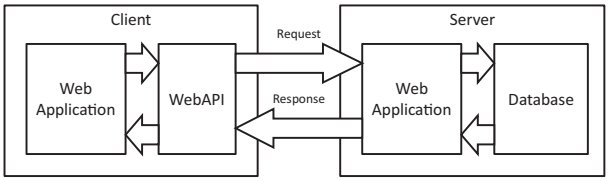

図 1: クライアントとサーバの構成

提案手法のクライアントライブラリはサーバの Webアプリケーションのある URLに対してリクエストを発行し，サーバの Webアプリケーションの処理結果を受け取っている．この時の処理結果はサーバの Webアプリケーションが実行された時に生成された HTML文書である．

2.2 提案手法の実装

提案手法の試作システムとして，クライアントライブラリを Ruby 1.9及び Mechanize 2.5.1[3] で実装した．試作システムは取得した Pixiv[2] の HTML文書を Rubyで扱えるデータ形式に加工している．試作システムの実装におけるクラス構成は図 2のようになっている．

Coreはライブラリ利用者が呼び出すことのできるクラス群である．Coreはサーバの HTML文書を取得し，Presenterのインスタンスをライブラリ利用者に返す役割を担っている．HTML文書は生成された Presenterに渡される．

Presenterは Rubyで扱えるデータ形式をライブラリ利用者に提供するためのクラス群である．生成された Presenterのインスタンスはサーバにおける 1つの HTML文書に相当する．Presenterは値が参照された

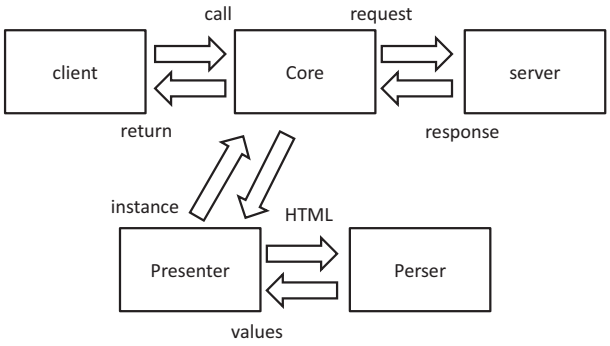

図 2: 試作システムの構成

際に Parser を呼び出し，HTML 文書の加工された値を返す．

Parser は Presenter から渡された HTML 文書を加工するためのクラス群である．Parser では HTML 文書を XPath[4] によって，指定した文書構造における文章から必要な値を取得，生成する処理を行う．

3 性能評価実験及び考察

本章では試作システムの性能評価実験とその考察について述べる．

3.1 性能評価実験

性能評価実験では，試作システムと Web ブラウザで Pixiv にアクセスし，リクエスト件数とレスポンスのデータ量について比較を行った．実験では Web ブラウザは FireFox 17.0.1 を利用し，データの計測には FireBug 1.11.1 を利用した．

表 1: リクエスト件数の比較（件）

	user	list	info	illust	合計
Web ブラウザ	90	61	42	3	196
試作システム	0	2	0	1	3

表 2: レスポンスのデータ量の比較（KByte）

	user	list	info	illust	合計
Web ブラウザ	745	144	196	268	1353
試作システム	0	16	0	266	282

表 1 及び表 2 の行の項目はそれぞれ Pixiv における Web ページであり，画面が遷移する順に並んでいる．member.php，member_illust.php，パラメータとして illust_id を追加した member_illust.php，パラメータの mode に big を指定した member_illust.php，それぞれの合計値となっている．ページ遷移は合計を除き，項目の左から右へ遷移する．

3.2 考察

表 1 では，Web ブラウザより試作システムのほうが約 65.3 倍もリクエスト件数が削減された．Web ブラウザで表示される Web ページには画像や CSS，Javascript のほかに広告なども含まれている．試作システムはこれらを取得しないことによってリクエスト件数を削減している．

表 2 では，Web ブラウザより試作システムのほうが約 4.7 倍もレスポンスのデータ量が削減された．illust では大きな画像を取得するためデータ量が多くなっているが，これを除いた場合は約 67.8 倍となる．Web ブラウザで表示される Web ページでは，レスポンスにサムネイルや広告の画像など，データ量の大きなファイルも含まれている．これらに対してリクエストを送っていないため，相対的にレスポンスのデータ量が削減されている．

4 おわりに

本論文では提案手法として，Web アプリケーションの HTML 文書を取得するためのクライアントライブラリについて述べた．提案手法の試作システムによって Web ブラウザよりもリクエストの回数とレスポンスで得られるデータ量が大幅に削減されたことが確認できた．

参考文献

[1] 山本裕介 (2012)：Twitter API と開発者規約変更のインパクトまとめ，http://www.atmarkit.co.jp/ait/articles/1209/26/news120.html, ITmedia (2012/12/21)

[2] Pixiv, http://www.pixiv.net/, 株式会社ピクシブ (2013/01/10)

[3] Michael Neumann：mechanize-2.5.1 Documentation, http://mechanize.rubyforge.org/, Mechanize (2013/01/11)

[4] J Clark, S DeRose：*XML path language (XPath) version 1.0* W3C Recommendation 16 November 1999