社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－ＤＤ－56（12）
2006／7／28

モバイルFeed統合ツールの開発（iアプリ版）

齊藤 秀治† 武内 孝憲† 力宗 幸男†

† 兵庫県立大学大学院 応用情報科学研究科 〒650-0044 神戸市中央区東川崎町 1-3-3
E-mail: †{aa04e203,aa05h209,rikiso}@ai.u-hyogo.ac.jp

あらまし Web1.0時代ではユーザ自らがWeb上をクロールし，情報を集めていた．しかし，Feedの登場で，より効率的に情報を閲覧できるようになった．また，総表現者時代と言われる今日，情報を受信するだけでなく，Blog等によって自ら情報を発信していく人々も増えてきている．そこで筆者らは，隙間時間にも情報発信・受信が行えるように，ケータイで利用可能なFeed統合ツールの開発を行った．

キーワード ケータイ，Blog，Feed/Atom Feed

A Feed Integration Tool for Mobile Terminals (i-appli version)

Hideharu SAITO†, Takeuchi TAKANORI†, and Yukio RIKISO†

† Graduate School of Applied Informatics, University of Hyogo Higashi-Kawasaki-cho 1-3-3, Chuo-ku , Kobe , 650-0044 Japan
E-mail: †{aa04e203,aa05h209,rikiso}@ai.u-hyogo.ac.jp

Abstract In the Web 1.0 age, you had to do web surfing to collect information from the web. But the appearance of the feed made the way of collecting information easier and more effective. Nowadays, in Web2.0 age, more and more people are not only getting information from blogs, but also writing information on blogs. We created a feed integration tool (i-appli version) with which the users can write to and read from several blogs using mobile terminals.

Key words mobile phone, Blog, Feed/Atom Feed

1. はじめに

『What is Web2.0 [1]』が発表されて，1年以上が経過した．2005年10月には『Web2.0カンファレンス』が開催され，Web2.0という言葉は世界中に拡散した．それらは先進的なビジネスパーソンや研究者たちを夢中にさせるのに十分な魅力をもっており，彼らの手によって様々な解釈が施されている．ではWeb2.0とは何か，という質問に対し，上原仁氏は『ウェブをプラットフォームとして位置付ける，オープン志向・ユーザ基点・ネットワークの外部性といった，インターネット本来の特性を活かす思想に基づいて提供されるサービスの次世代フレームワーク [2]』と定義できるのではないかと答えている．

それは，WWWの発明者ティム・バーナーズ・リーがセマンティックウェブの基本概念として語った『Anyone can say anything about anything』という言葉に酷似している．

では，その言を体現したWeb2.0のサービスは何か，との問いに対して，おそらく大半の人がBlogを思い浮かべるではないだろうか．

そこで本研究では，誰もが気軽にWeb上で自己表現することのできるサービスであるBlogに注目し，Blogをさらに有効活用できるようなシステムの構築を行った．そして，プラットフォームに依存しない，且つ場所を選ばないサービスというものを念頭に置き，ケータイ(注1)で利用するアプリの構築を行った．前述のバーナーズ・リーの言葉に数語を付け加え，『Anyone can say anything about anything In Anywhere』の概念を基に本研究は臨んでいる．

2. Blogとは

Blog(注2)とは，簡易ホームページと表現されるCGM(注3)の一種である．

Blogの特徴としては

- 誰でも容易に自サイトを開設できる
- 構造的なWebサイトを構築できる

(注1)：今回はdocomoを対象
(注2)：WeBlogともいう
(注3)：Consumer Generated Media．情報媒体（メディア）の中でも，消費者（コンシューマー）が自ら情報を発信するメディアの総称．

- Feed/Atom Feed を自動で生成できる

等が挙げられる．

元々，Blog はコンシューマ向けのサービスとしてスタートしたが，上記のような特徴が SEO[注4] 対策として適していたため，ビジネス分野でも幅広く採用されてきており，現在では Blog が Web サイトの基盤となりつつある状況である．

3. Feed/Atom Feed とは

元々，Feed/Atom Feed は，Web サイトのメタデータとして更新情報を簡易かつシステマティクに取得できる点が評価され，プッシュ型ライクなプル型配信として利用されてきた．だが，現在では，単なるメタデータで終わらず，さらに一歩進んだ利用方法が考えられ始めている．

Web2.0 という言葉が世に出てから既に 1 年以上が経過し，いくつものサービスを組み合わせることで新たなサービスを構築するマッシュアップという手法が一般化し始めている．だが，各種サービスがマッシュアップにより互いに連携するためには共通のデータフォーマットが必要であり，その役割を Feed が担おうとしている．フィードパス社の小川氏の言を借りれば『Feed が Web2.0 の血液となる』わけである．

後述する Atom API においても，Feed をデータフォーマットとした API を利用することで，マッシュアップ的にサーバサイドシステムを構築している．

4. Feed リーダ・ライタについて

Feed リーダ・Feed ライタという表現は，まだ一般的ではない．現在においてユーザが Feed を利用する場合，その対象は Blog であることがほとんどであり，名称としては Blog リーダ・Blog ライタの方が適切ではある．しかし前述のように，『Feed が Web2.0 の血液となる』ことで，Feed の利用は Blog に限らなくなると思われる．それら将来性を見越した上で，本研究では Feed リーダ・Feed ライタ[注5] という表現をとっている．

4.1 Feed リーダ

Feed はあくまでも XML であり，人間がそのまま閲覧するには適していない．そのため，HTML 文書を読むために利用するアプリケーションである Web ブラウザに該当する，Feed リーダ[注6] というアプリケーションが必要である．

その Feed リーダであるが，大別して以下の 4 種類に分類する事が出来る．

- アプリケーション・メーラ型
- アプリケーション・ティッカー型
- ホスティグ型
- モバイル型

それぞれの特徴を以下に記す．

4.1.1 アプリケーション・メーラ型

クライアント PC にインストールして利用するタイプの Feed リーダ．UI がメーラに似ており，また後述のホスティング型よりも動作が機敏が事が多いため，初心者ユーザでも容易に利用できる．

glucose[注7] などが代表的．

4.1.2 アプリケーション・ティッカー型

クライアント PC にインストールして利用する分には，上記のメーラ型と同様だが，デザインが電光掲示板型．このため，別作業をしながらニュースのヘッドラインを流し読みする際に有用である．

e クルーザー[注8] が代表的．

4.1.3 ホスティグ型

Web ブラウザを利用して，Feed を閲覧する Web アプリケーション．Web アプリケーションなので動作が鈍いという問題点があったが，Ajax を利用する事により，クライアントアプリケーションと同レベルの操作性を持った Feed リーダも登場してきている．

Feedpath[注9]，Livedoor Reader [注10] などが代表的．

4.1.4 モバイル型

ケータイや PDA で利用する事を主目的として開発されている Feed リーダ．移動時等の隙間時間に利用できる，これから発展が望まれるタイプ．

ECR Feed Reader[注11] が代表的．

4.2 Feed ライタ

Feed ライタとは，Blog の管理画面からエントリを投稿するのではなく，独自のサービスを付加した形のエントリを投稿できる，サードパーティ的なアプリケーションである．

その Feed ライタであるが，大別して以下の 3 種類に分類する事ができる．

- クライアント・アプリ型
- ホスティング型
- ケータイ型

それぞれの特徴を以下に記す．

4.2.1 クライアント・アプリ型

クライアント PC にインストールすることで利用可能なアプリケーション．インストール型のため動作は機敏であるが，投稿先 Blog のシステム変更があった場合，ユーザは再度ダウンロード・インストールをしなければ対応できない．

ecto[注12] や glucose などが該当する．

4.2.2 ホスティング型

Web ブラウザを利用して，Blog に投稿する Web アプリケー

(注4)：Search Engine Optimization．サーチエンジンの上位に自分の Web ページが表示されるように工夫すること．また，そのための技術．
(注5)：本システムではリーダ・ライタ機能を併せ持つため，Feed 統合ツールと呼称する
(注6)：アグリゲータともいう
(注7)：http://glucose.jp/
(注8)：http://www.redcruise.com/
(注9)：http://feedpath.jp/feedreader/
(注10)：http://reader.livedoor.com/
(注11)：http://www.ecreal.co.jp/pc/top/index.html
(注12)：http://ecto.kung-foo.tv/index.php

ション．Webアプリケーションであるため，ユーザにインストールの手間を省かす事ができたり，Web上の様々な情報を利用する事でユニークな付加価値を付け加えることを可能としている．

Feedpathや nAmazlet(注13)がそれに該当する．

4.2.3 ケータイ型

ケータイのアプリ機能を利用する事で，Blogに投稿するアプリケーション．ケータイを用いる事で隙間時間にもBlog投稿を可能としている．

本システムが日本で初めて(注14)構築した．

たしかに，ケータイからのBlog投稿で同様のサービスとしては，モブログというメールを利用した投稿サービスが存在する．ただし，あくまでも投稿するのみであり，そこに付加的なサービスを挟み込むことはできない．そのように，メールによる投稿は簡便ではあるが，自由度が低くなってしまう．

だが，本システムでは中間処理を挟む事で，様々な付加サービスを実現可能である．また，ケータイのアプリ機能を利用する事で，メール機能では実現できないUser Experiencesを実現している．そのように，ケータイから自由度の高い投稿を可能にする事で，移動時などの隙間時間にBlogの投稿を可能としている．

5. システム概要

今日では，前述したように自分のホームページの代わりになり，簡単に作れるBlogが爆発的な広がりを見せており，数多くのBlogサービスが世に出回っている．Blogは世相や時事問題，専門的話題に関しての独自の情報や見解を掲載するといった情報発信の場となっている．また，自動的にFeedを生成してくれるために，Feedリーダを用いることで情報収集が容易となっている．

本システムはケータイを用いてBlogに記事（情報）を投稿（発信）でき，かつFeedによる情報収集（受信）できるシステムである．

5.1 システムの構成

本システムは通信を主体としたシステムであり，サーバサイドとアプリサイドの2つのシステムから構成されている．

表1 本システムの構成

	本システム	
システム	サーバサイド	アプリサイド
言語	PHP5.0	Doja3.5
主な機能	1.Blogサービスとの通信 2. ケータイとの通信 3.DBにFeedを登録	1.Blogへの投稿，編集，削除 2.Feed，HTMLの取得，表示 3. 各設定データの保存，削除

(注13)：http://www.siu-labo.net/app/hreview/

(注14)：海外をみれば，本システムの機能制限板とも言えるPico Stationというサービスがある

サーバサイドではPHP5.0を用い，アプリサイドではNTTDoCoMoから提供されているDoja3.5(注15)を用いて開発をした．本システムはFOMAの90Xシリーズで利用可能である．

5.2 システムの処理の流れ

本システムではアプリの容量を軽減させるために，サーバサイドで多くの処理を行っている．投稿する際はケータイの方から直接Blogサービスに送るのではなくて，一旦サーバの方で処理を行ってから送る，という形をとっている．Feedを取得する際はその逆の形である．アプリの方にFeedを登録するには，サーバにあるDBから選択する方法と，直接URLを手入力する方法があり，アプリの方に登録するとDBの方にも登録される．これによってユーザが増えれば増えるほど，多くのFeedから選択できるようになっている．図1に処理の流れを示す．

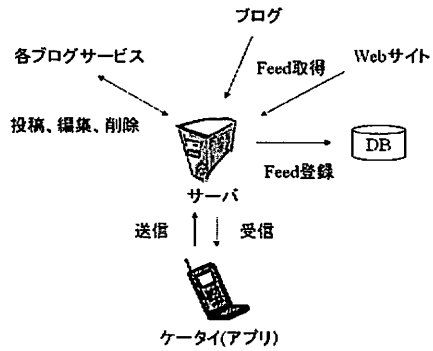

図1 システムの処理の流れ

6. システム詳細

6.1 サーバ詳細

サーバサイドで利用している技術としては，XML-RPC API[3]とAtom API[4]である．

また，両APIを簡便に扱うための独自ライブラリおよびクッションAPIを構築し，ケータイアプリからAPIを呼び出す際のデータフォーマットの違いなどを吸収するようにしている．

6.1.1 XML-RPC API

SOAPの基となった仕様で，インターネット上でリモートプロシージャコールを実行するためのプロトコル．クライアントがXML形式のテキストで記述された型付の引数をサーバ側アプリケーションに渡し，サーバが返り値を同じくXML形式のテキストで返すという動作をする．

6.1.2 Atom API

正式な名称は，The Atom Publishing Protocol.

HTTPのGET/POST/PUT/DELETEを特定のEnd-PointURLに対して行い，そのリクエストに規定のXML文書

(注15)：NTTDoCoMoのiアプリ用のJava拡張ライブラリ．J2ME CLDCの拡張ライブラリで，スクラッチパッドや携帯電話のボタンの処理機能などが定義されている．ver1.0,ver2.0,ver2.1,ver3.0,ver3.5,ver4.0,ver4.1が提供されている．

を加えて送信することでインタフェースが用意している操作を行うことができる．また，一部の操作はそのレスポンスとして規定 XML 文書を返却する．その際に送受信される XML 文書は Atom フォーマットに基づいている．

6.1.3 クッション API

XML-RPC API も Atom API も XML データを EndPointURL に一定の手続きに従ったデータフォーマットを送信する事で，Blog アクション(注16)をおこす．このデータフォーマットであるが，XML-RPC API・Atom API ともに XML 形式のデータフォーマットであり，API 制御のための冗長的なデータも含まれている．スペック制限の受ける事の少ない PC・ネットワーク環境であれば，冗長的な部分を含むデータを送受信しても問題はないが，本システムはケータイアプリであるために，CPU・ネットワークともに制限を受けてしまうため，できる限り送受信するデータは小さい方が好ましい．

そこで，本研究では，XML-RPC と Atom とを仲介する独自ライブラリ『BlogWriteAPI』利用したクッション API を構築した．

この API は XML-RPC API と Atom API を REST 化するものであり，ケータイからこの API にデータを POST する．すると，受け取ったデータをサーバ側で XML 形式に変換し，それを XML-RPC API や Atom API として投稿先 Blog の EndPointURL に送信する．

また，レスポンスとして返ってきた XML データをパースし，ユーザにとって必要な部分だけをケータイに返している．

このように，ケータイからの送受信で使うデータを XML フォーマットではなくしたことで，送受信領域に限界があるモバイル通信においても冗長的なデータを送らずに済み，一定の通信速度向上に役割を果たしている．

また，ケータイから直接 XML-RPC API や Atom API を利用して Blog アクションを起こすのではなく，中間にクッション API を挟んで Blog 投稿する事により，XML-RPC API や AtomAPI に対応していない Blog に対しても Blog アクションを起こす事が可能となる．

また，投稿プロセスを開発者の管理下をおく事で，様々な付加価値サービスを追加していく事ができる(注17)．

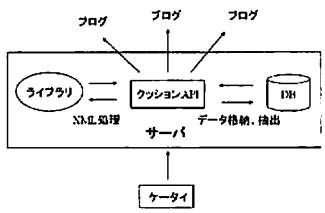

図 2 サーバの処理の流れ

(注16)：エントリの投稿，編集，削除など

(注17)：現在は付加価値としてのタグ機能を提供

6.2 アプリ詳細

本システムのアプリサイドは「Blog ライタ」と「Feed リーダ」の二つから構成されている．この二つの機能を併せ持つことによって，Feed リーダを用いて情報を取得し，得た情報に対しての記事を，Blog ライタを用いて投稿することで情報を発信することができる．

6.2.1 Blog ライタ

Blog ライタを使用する前提として，ブログアカウントを取得済みでなければならない．本システムではブログアカウントを取得することができないために，あらかじめユーザに行ってもらう必要がある．投稿するブログサービス先とそのアカウント，パスワードを入力し，認証で問題がなかった場合，Blog ライタを使用することができるようになる．図 2 は Blog ライタのトップ画面である．

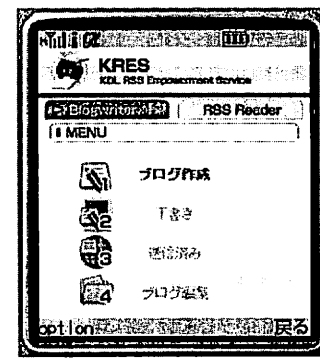

図 3 Blog ライタのトップ画面

6.2.2 Blog ライタの機能

Blog ライタの主な機能は以下の 4 つである．

- ブログ作成
- 下書き
- 送信済み
- ブログ編集

「ブログ作成」は，あらかじめ登録しているブログサービスを選択し，ブログの題，本文，タグ(注18)を入力して送信（投稿）する．メールを打つのと同じ感覚で利用できる．また，テキストだけではなく画像を貼り付けて送信することも可能であり，すでにケータイに保存されている画像から選択する場合と，アプリからカメラを起動して，撮った画像を貼り付ける場合の 2 通りがある．図 3 はブログ作成画面である．一度に送れる画像サイズには制限があるため，それを超える画像サイズの場合は分割して送るようにしている．

「下書き」は「ブログ作成において書いた記事を一時保存しておく機能であり，ケータイが通信できない環境にあるときなどに用いる．

(注18)：現在，テクノラティタグを利用している．

「送信済み」は送信した記事が保存されており，どんな記事を送ったかを確認することができる．また，送信した記事を編集，削除することができるために，PC からではなく，ケータイの方から送った記事を書き直したい場合や，消したい場合などに用いる．

「ブログ編集」は登録しているブログサービスの記事を編集，削除することができる．前述したように「送信済み」からでも編集はできるが，このアプリを使って送信した記事だけでなく，PC（ブラウザ）から送信した記事も編集，削除が可能となっている．

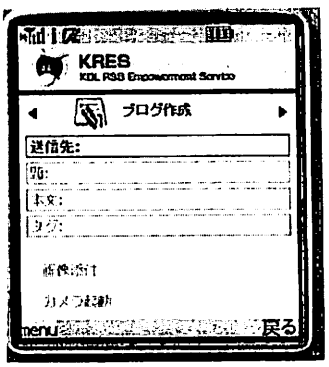

図 4　Blog ライタのブログ作成画面

6.2.3　Feed リーダ

Feed リーダの方は Blog ライタとは違って，すぐに使用することができる．デフォルトではアプリに関するお知らせの Feed，アプリの使い方の Feed，また NEWS サイトの Feed が 5 件，全 7 件が登録されており，この 7 件は削除できないようになっている．

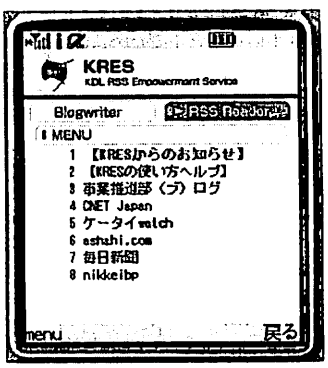

図 5　Feed リーダのトップ画面）

6.2.4　Feed リーダの機能

Feed リーダの主な機能は以下の 4 つである．

- Feed (title,description,date)，URL の取得
- HTML の取得
- Feed 登録
- Feed リーダ，Blog ライタ連携

「Feed の取得」は title,description,date の 3 つを最新のものから 20 件取得し，title,date を表示する．読みたい Feed を選択すると description が表示される．title が長い場合はティッカーとして読めるようになっている．また全文の HTML が読めるように，その URL も取得している．

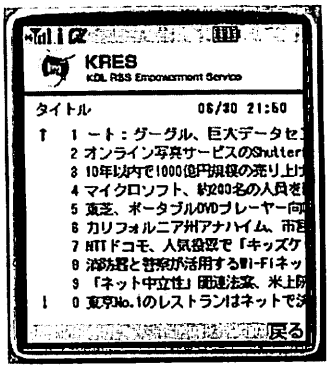

図 6　Feed リーダの Feed 表示画面

「HTML の取得」は Feed 表示画面で，概要だけでなく更に詳しい情報を読みたい場合に利用する．サーバで HTML の < a > タグなどを省く処理を行い、省いたものを受け取り，表示する．

「Feed 登録」は登録したい Feed の URL を直接手入力する「URL 入力」と，サーバにある DB から興味があるラベル(注19)を選び，そのラベルと合致する Feed を選択し登録する「ラベル検索」の 2 つから行うことができる．

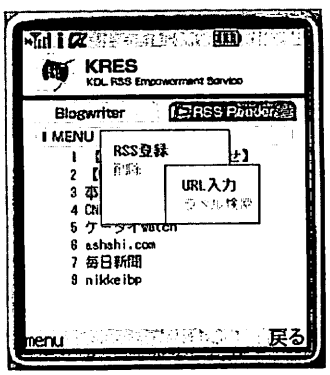

図 7　Feed リーダの Feed 登録画面

「Feed リーダ，Blog ライタ連携」とはリーダで読んだ HTML に対して，ライタで投稿することができる機能である．これは，HTML の URL を < a > タグで囲んだテキストをブログ作成画面の本文に自動的に入力する機能と，読んだ HTML を引用し

(注19)：ラベルとはタグと同じ意味であり，ライタの方と区別するためにラベルと命名している

たい場合にケータイのコピー機能を使って，引用したい部分をコピーし，コピーしたテキストをブログ作成画面の本文に自動的に入力する機能の2つから選択できるようになっている．

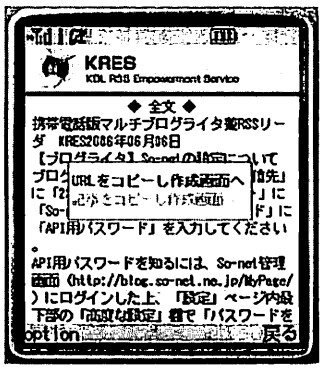

図8 Feed リーダ，Blog ライタ連携の画面

6.2.5 アプリのインターフェース，ユーザインターフェース

インターフェースは，各端末ごとで見栄えが変わらないように開発しており，インターフェースを統一する，かつ機種依存の影響を受けないように Panel 型ではなく，Canvas 型を使用し，文字の大きさも 12 ドット＊ 12 ドットで統一している．

ユーザインターフェースは「ユーザが使いやすい」という点を重視して開発しており，主な以下の 3 点などがある．

- マルチスレッド利用
- ショートカット機能
- アカウント，Feed のダブル登録
- 容易な画面遷移

「マルチスレッド」(注20)を用いているため，通信中でも通信はバックグラウンドで行うため，アプリ操作が可能となっている．例えば 3 件下書きに貯めている記事を送るときには，1 件目の通信が終わるのを待ってから 2 件目を送るのではなく，待たずに連続して送る，といったことが可能である．

「ショートカット機能」とは選択項目の左についている番号と同じ番号をケータイで押すと，その番号の項目が選択されるようになっている．

「アカウント，Feed のダブル登録」とはアカウント登録をしたときに，そのブログ先の Feed の登録を任意で選べるようになっているため，後で自分のブログ先の Feed の URL を手入力する，といった面倒な作業は省けるようになっている．

「容易な画面遷移」とは Blog ライタおよび Feed リーダのトップ画面時に，ケータイの左キーまたは右キーを押すことで，Blog ライタから Feed リーダ，Feed リーダから Blog ライタへと簡単に画面遷移ができるようになっている．ライタ使用時においてブログ作成画面，下書き画面，送信済み画面，ブログ編集画面の場合に左右キーを押すことで，画面遷移ができるようになっている．また，Feed リーダでの description および HTML の表示画面において，右キーを押すと 1 ページ下にスクロールし，左キーで 1 ページ上にスクロールするようになっている．

表2 画面遷移

使用画面	左キー	右キー
Blog ライタトップ画面	RSS リーダトップ画面へ	RSS リーダトップ画面へ
RSS リーダトップ画面	Blog ライタトップ画面へ	Blog ライタトップ画面へ
ブログ作成画面	ブログ編集画面へ	下書き画面へ
下書き画面	ブログ作成画面へ	送信済み画面へ
送信済み画面	下書き画面へ	ブログ編集へ
ブログ編集画面	送信済み画面へ	ブログ作成画面へ

7. 結論・まとめ

本研究では，Web2.0 の流れを受けて本システムを開発した．本システムではケータイを用いて使用するため，ケータイの最大メリットである「いつでも，どこでも」情報発信，情報収集ができるようになった．今日では，ケータイは 1 人に 1 台と言っても過言ではないくらいに普及しており，より多くの人に使用してもらいたいと考えている．ただ，現段階では NTTDoCoMo の FOMA90X シリーズにしか対応していないため，今後は多くの端末で使用できるようにするつもりである．

今後の課題として次のようなものが挙げられる．

（1） DoCoMo だけでなくマルチキャリアに対応
（2） FOMA70X シリーズにも対応
（3） 機種依存によるバグの修正
（4） 登録している Feed のフォルダ管理
（5） Feed の未既読管理

さらに利用しやすいものとするために，実際にユーザーに利用してもらい，よりユーザインターフェースの良いシステムへと改善することが必要である

文　献

[1] Tim O' Reilly,『What is Web2.0（和訳版）』CNET Japan（http://japan.cnet.com/column/web20/story/ 0,2000054679,20090039,00.htm），2005
[2] インターネットマガジン『WEB2.0 への道』内記事『新潮流!Web2.0』インプレス出版,2006
[3] XML-RPC 仕様書 http://lowlife.jp/yasusii/stories/9.html
[4] Atom Publishing Format and Protocol (atompub) Charter http://www.ietf.org/html.charters/atompub-charter.html
[5] i アプリコンテンツ開発ガイド for DoJa-3.0　詳細編 http://www.nttdocomo.co.jp/binary/pdf/service/imode/make/content/iappli/about/jguideforDoJa3_0_040428.pdf
[6] i アプリコンテンツ開発ガイド for DoJa-3.5 API リファレンス編 http://www.nttdocomo.co.jp/service/imode/make/content/download/kiyaku3.html

（注20）：1 つのアプリケーションソフトがスレッドと呼ばれる処理単位を複数生成し，並行して複数の処理を行なうこと．いわばアプリケーションソフト内でのマルチタスク処理のこと