モバイル・デバイス向けマッシュアップ環境構築のアプローチ

神山 淑朗　野口 雅人

An Approach for Building a Mashup Environment for Mobile Devices

Yoshiroh Kamiyama and Masato Noguchi

次世代ネットワークの登場により，通信とITを融合した新しい形態のサービスを実現する基盤が整いつつある．加えて高速データ通信や高機能ブラウザーを備えるスマートフォンの登場で，それらのサービスをモバイル・デバイス上で利用することも可能になる中で，通信事業（テレコム）においてもWeb 2.0のマッシュアップの概念が急速に取り入れられようとしている．本論文では，筆者らの考えるテレコム・マッシュアップの要件と課題を整理し，その解決策と実現方法について述べる．特に受信通知と遅延ロードという本来相いれない要件に対するアプローチを提案し，実際の環境構築を通して機能を検証する．さらに，本提案手法はコンポーネントの再利用性および開発ツールへの親和性の観点でも有効であることを示す．

A Next Generation Network (NGN) provides an infrastructure on which new types of services that leverage the convergence of telecommunications and information technology can be built. In addition, the advent of smartphones that offer high-speed wireless data access and modern browsers enables such services to be used on mobile devices, thereby making it possible to apply the Web 2.0 mashup concept to telecommunications. In this paper, we describe our “telecom mashup” concept, identify its requirements and problems, and present our solutions to these problems and our implementation method. More specifically, we propose an approach which makes it possible to realize both event notification and lazy loading, which had generally been considered to be incompatible, and evaluate our approach through actual implementation of a telecom mashup environment. Furthermore, we demonstrate that the proposed approach is also beneficial to component reusability and development tool friendliness.

Key Words & Phrases：Telco 2.0，Web 2.0，次世代ネットワーク，通信事業，マッシュアップ，モバイル・デバイス
Web 2.0, Next Generation Network, telecommunication, mashup, mobile device

1. はじめに

携帯可能な情報端末であるモバイル・デバイスの普及が進み，社会や生活の中に広く浸透している．特にその代表格である携帯電話は，すでに市場が飽和状態を迎えつつあるために，通信事業者は既存ユーザーへ付加価値を提供することで，より収益の高い新しいビジネス・モデルを模索している．そのような中，いつでもどこでもインターネットに接続でき，デスクトップPCに匹敵する能力を持つスマートフォンが急速な勢いで成長を見せている．2010年1月のガートナー調査［1］でも，2013年までにスマートフォンのようなモダン・ブラウザーを搭載した高機能携帯電話が，デスクトップPCを超えて全世界で最も一般的なWebアクセス手段となるであろうと報告している．SaaS（Software as a Service）のような実用的なサービスからSNS（Social Networking Service）のような魅力的なサービスまで，Web上のあらゆるサービスがモバイル・デバイスから活用できる時代に入ったといえる．

一方，付加価値の高いサービスを提供したい通信事業者は，電話網をオールIP化する次世代ネットワーク（NGN:Next Generation Network）［2］［3］の商用サービスをスタートさせた．NGNは，従来の電話網が持つ信頼性・安定性を確保しつつ，IPネットワークのオープン性，柔軟性，経済性を備える．これにより，従来は外部からの利用は不可能であったコール制御，メッセージング，課金などの通信事業者による各種サービス（テレコム・サービス）を，Webの世界にAPI（Application Programming Interface）として公開することが可能と

提出日：2009年5月11日　再提出日：2010年6月4日

なった．Web にサービスを公開するための技術として Parlay X［4］と呼ばれるオープン・スタンダードなテレコム・サービス用の API が策定・実装され，Web サービス経由でテレコム・サービスが利用できるようになった．IT 業界で培われた IP ベースの技術を活用し,オープンなサービスや多彩なアプリケーション，さらにはそれらを組み合わせたさまざまな複合サービスをモバイル・デバイス・ユーザーに対して提供することを可能にするインフラが整ったといえる．

テレコムの業界におけるオープン化の流れが現実のものとなるにつれ，Web 2.0［5］［6］の原理を取り込もうとする動向がより鮮明になった［7］．オープンな技術でテレコム・サービスの API を公開し，インターネット上のほかのサービスや企業をはじめとするさまざまなサービス提供者のサービスと組み合わせて（マッシュアップして），ユーザーに新たな価値を生む創造性を与え，コミュニティーあるいはエコシステムを築き,集合知を活用する，といった考えである．これは，Web 2.0 のテレコム版，あるいは，テレコム・サービスの価値を付加した Web 2.0 のアプローチという意味で「Telco 2.0」と呼ばれている［8］．

Telco 2.0 では，契約者向け音声通話サービスという従来のモデルから，オープンなサービスを柔軟に組み合わせて活用するというモデルへパラダイムシフトが起きた．すでに PC 上の Web2.0 のプラットフォームにテレコム・サービスを組み込んで活用する試みは行われているが［9］，それをモバイル・デバイス上で実現しようとすると，パフォーマンスの向上や UI（ユーザー・インターフェース）の工夫といったモバイル特有の課題を解決する必要が出てくる．

本論文では，テレコム・サービスやほかのサービスをコンポーネント化し，それらのコンポーネント（=ウィジェット）をマッシュアップすることで，エンド・ユーザーが独自のアプリケーション（シチュエーショナル・アプリケーション）を作成できる手法と，それをモダン・ブラウザー搭載のモバイル・デバイス上で利用できるシステムについて述べる．また，アプリケーション起動時のパフォーマンスを向上させるためには，コンポーネントの遅延ロードは有効な手段ではあるが，一方でコンポーネントが何らかのメッセージを受信したことをユーザーに知らせるためには，すべてのコンポーネントはあらかじめロードされていなければならず，この両者の相反する要件を両立させるための仕組みを提案する．

以下，2 章で，モバイル・デバイス向けのマッシュアップの意味と意義の解説を行い，3 章で，その要件と課題を整理し，4 章で，解決策のアイデアを述べる．次に，5 章で，具体的なテレコム・マッシュアップ環境構築例の紹介を行い，6 章と 7 章で，実装の結果を考察する．

2. モバイル・デバイス向けマッシュアップとは

サービスをマッシュアップし，さらにそれをモバイル・デバイス上で利用するとはどういうことか，その意味および想定されるシナリオを整理する．

2.1 マッシュアップの必要性

複数のサービスやアプリケーションを組み合わせて，新しいアプリケーションを作るマッシュアップは，欲しい時に欲しい機能を利用者自身の手で実現するシチュエーショナル・アプリケーションの作成に利用される．従来型の作り込み Web アプリケーションと比較した場合，マッシュアップの特徴は次の点にある．

（1）一般のビジネス・ユーザー（= 非 IT 専門家）が自分用のアプリケーションを自分で組み立てることができる．
（2）対象アプリケーションは 1 度作ったらずっと使い続けるような性質のものではなく，常に改変，改良，カスタマイズ，新規作成を繰り返すものである．

これらの特徴を必要とするシナリオではマッシュアップの効果が最大限に発揮される．逆にいうと，これらの特徴を必要としない場合，すなわち，IT の専門家が開発したアプリケーションをそのまま使い続けるような状況の場合は，エンド・ユーザーにとってはマッシュアップの必要性は低下する．

2.2 モバイル・デバイス上で利用する意味

国際標準の API 仕様である Parlay X を使用することで，アプリケーションから各種テレコム・サービスへアクセス可能になり，さらにそのアプリケーションをモバイル・デバイス上のブラウザーから利用することも可能である．ここで，モバイル・デバイス上でテレコム・サービスを利用することに関して若干の疑問が生じるかもしれない．例えば，代表的なサービスの 1 つは 2 者間の呼を設定するサードパーティー・コールであるが，携帯電話上で動作する電話をかけるアプリケーションというのは一見奇妙に思える．しかし，実際には単にユーザーが手動で電話をかけるだけにとどまらず，サービスの利用の仕方は完全にプログラマブルである点に注目したい．従って，例えば，株価があるしきい値に達したら自動的に投資家とブローカーを電話でつなぐといったよう

に，アイデア次第で多様なサービスやアプリケーションを提供できる可能性を秘めているといえる．

2.3 シナリオ

今回筆者らの構築するシステムでは，前述の点を踏まえ，次のようなシナリオを設定した．

「ビジネス･ユーザーが，出張するに当たり，前の晩に，旅先で必要になりそうなウィジェットを自分の PC で配置･連携設定してアプリケーションを準備しておき，出張先でそれをモバイル・デバイスで利用する」

このようなシナリオでは，自分の出張用に特化したアプリケーションを自分で用意すること，出張目的に応じて毎回作り直すこと，といった状況から，マッシュアップが効果を発揮する．

3. モバイル･デバイス向けマッシュアップの要件と課題

3.1 想定される要件

テレコム・サービスも含めた多様なサービスを組み合わせて活用するコンポーネント・ベースのマッシュアップ・アプリケーションをモバイル・デバイスで快適に利用可能にするには，次の 3 つの要件を満たす必要があると考える．

(1) モバイル・デバイスは画面が狭いため，各コンポーネントをアイコンや折り畳み可能なセクションなどの形で配置し，必要なときだけ開くようにする：レイアウト・コンテナ．

(2) コンポーネント間連携により，あるコンポーネントが別のコンポーネントからメッセージを受け取った場合，またはデバイスやサーバーからイベントを受け取った場合などに，例え受け取ったコンポーネントが現在閉じられていてもユーザーにそれを通知する：受信通知（Notification）．

(3) モバイル・デバイスはネットワーク・CPU 共に PC ほどの能力はないので，起動時のパフォーマンス向上のために，マッシュアップ・アプリケーションの起動時に閉じられているコンポーネントはロードせず，コンポーネントが初めて開かれたときコードをロードし，インスタンス化する：遅延ロード（Lazy Loading）．

3.2 受信通知と遅延ロードの課題

受信通知と遅延ロードは，通常は両立しない要件である．なぜなら，メッセージの受信通知を行うためには，コンポーネントが自分あてのメッセージを受信しなければならず，そのためにはコンポーネントはあらかじめロードされ，存在していなければならないためである．つまり，遅延ロードによりコンポーネントがまだ存在しない段階では受信通知を行うことはできないという問題がある．

4. 受信通知と遅延ロードの両立のためのアプローチ

受信通知と遅延ロードの両立を実現するため，次のアプローチを考案した．

(1) コンポーネントを次の 2 つの部分に分け，二重構造とする．
- コンポーネント・スタブ
- コンポーネント本体

(2) アプリケーション起動時には各コンポーネントのスタブのみをロードし，実装コードの大部分を占める本体は必要になるまでロードしない（遅延ロード）．

(3) コンポーネントが外部からメッセージを受信すると，まだ本体がロードされていない場合は代理でスタブが受信する．スタブはユーザーにメッセージの受信を知らせる（受信通知）と共に，受信したメッセージ内容を記録しておく．

(4) ユーザーによってコンポーネントが開かれると，そこで初めてコンポーネント本体がロードされ，実行される．そのとき，記録しておいたメッセージをコンポーネント本体に届ける．

以下，本アプローチの詳細を述べる．

4.1 コンポーネントの構成

図 1 にコンポーネントの構成を示す．コンポーネントはコンポーネント本体およびそれを取り囲むコンポーネント・スタブの二重構造を成す．さらにコンポーネント･スタブは，コンポーネント・フレームおよびコンポーネント・ベースから構成される．

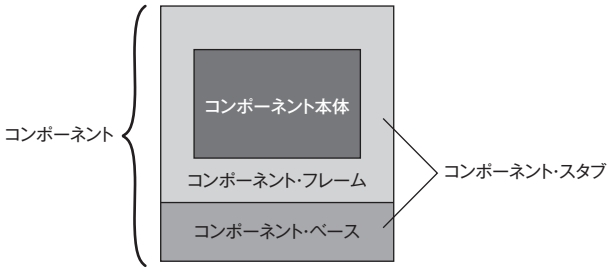

図 1. コンポーネントの構成

a) コンポーネント本体

コンポーネントごとに固有の実装コードであり，典型的には，ユーザー・インターフェース，サービス呼び出し，ビジネス・ロジックなどを含む．また，本論文ではマッシュアップ環境を想定しているため，ほかのコンポーネントと連携するためのメッセージ送受信の機能も備える．

b) コンポーネント・フレーム

コンポーネントの形式的な枠組みであり，コンポーネントごとに固有の定義を有する．具体的には，どのような種類のイベントを外部へ送信するかを示す「送信イベント定義」，どのような種類のイベントを外部から受信可能かを示す「受信イベント定義」，コンポーネント同士の接続を示す「コンポーネント間連携定義」である．

c) コンポーネント・ベース

全コンポーネントに共通するモジュールであり，次の機能を有する．

- コンポーネント本体のロード
- コンポーネント本体のインスタンス化
- メッセージの受信
- レイアウト・コンテナに受信を通知
- 受信メッセージの記録

d) コンポーネント・スタブ

コンポーネント・フレームとコンポーネント・ベースから成る外枠のみで中身のないコンポーネントであるが，外部からは通常のコンポーネントと同様に見える．それにより，コンポーネントを扱うツール（コンポーネント開発支援ツール，マッシュアップ・プラットフォームなど）や，コンポーネントのランタイム（実行環境）には特殊な作り込みを必要とせず，仮にコンポーネント本体が存在しなくても通常のコンポーネントとして扱うことができる．

4.2 レイアウト・コンテナ

レイアウト・コンテナは，表示・非表示の切り替えが可能な複数のペイン（表示領域）を持つ UI 部品である．ペイン内にコンポーネントを配し，表示・非表示の管理を行う．この種の UI としては，折り畳みセクション，アイコン，タブパネル，アコーディオンなど，さまざまなバリエーションが考えられる．

本論文におけるレイアウト・コンテナは，アプリケーションの起動直後に，図 2 のような構成を取る．すなわち，各ペインにはそれぞれコンポーネントが 1 つずつコンポーネント・スタブのみの状態で配置され，画面上では非表示状態となっている．

また，メッセージ受信通知を行うために，ペインを閉じた状態で変化を知らせる機能（例えば，折り畳みセクション・タイトルをハイライト表示する）を備える．

4.3 受信通知と遅延ロードの処理手順

ステップ 1：コンポーネント #1 を開く（図 3）

レイアウト・コンテナは，セクション・タイトル「コンポーネント #1」がクリックされたことを受け，コンポーネント #1 のコンポーネント・ベースが持つ createWidget() メソッドを呼び出す．createWidget() は次の処理を行う．

- コンポーネント本体の実装コードをロードし，インスタンス化を行う．
- メッセージ受信ルーチンを差し替える．

（旧）受信メッセージを記録し，レイアウト・コンテナに通知する．

（新）受信メッセージを本体に転送し，レイアウト・コンテナに通知する．

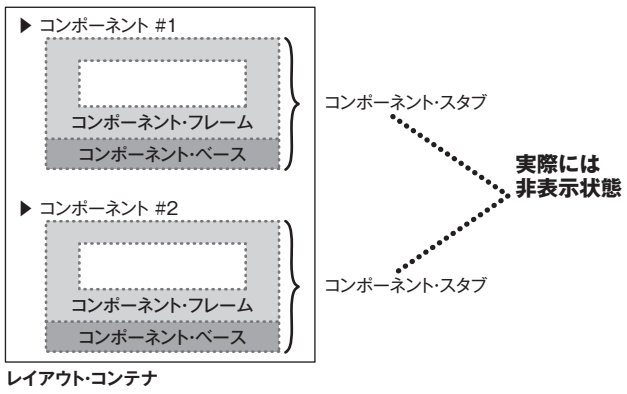

図 2. レイアウト・コンテナの構成

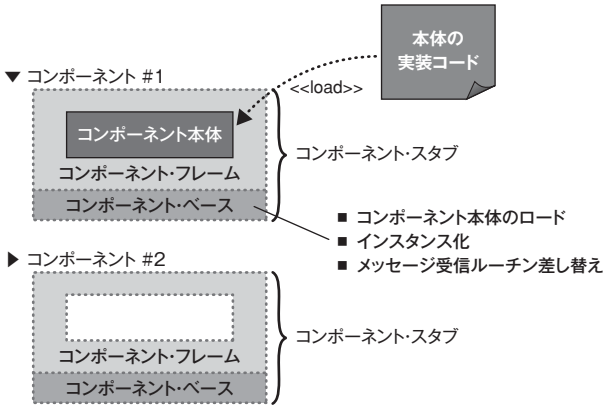

図 3. コンポーネントを開く

ステップ 2：メッセージを送受信する（図 4）

コンポーネント #1 の操作により，連携するコンポーネント #2 へメッセージが送信される．例えば，データ表示用テーブルの行を選択する操作により，選択行のデータがメッセージとして送信され，受信側はその行データを受け取る．

コンポーネント #2 のコンポーネント・ベースは次の処理を行う．

- メッセージを受信する
- レイアウト・コンテナに受信を通知する
  → レイアウト・コンテナはセクション・タイトルをハイライト表示する
- 受信したメッセージを記録する

（本来メッセージを処理すべき本体がまだ存在しないので後から処理できるよう一時的に保管しておく）

コンポーネント #2 の実体はまだ存在していないが,ユーザーに対するメッセージ受信の通知を可能としている．

ステップ 3：コンポーネント #2 を開く（図 5）

ステップ 1 と同様にしてコンポーネント #2 を開く．ただし，コンポーネント #2 がロードされる前に受信したメッセージが記録されている．コンポーネント #2 のコンポーネント・ベースは次の処理を行う．

- コンポーネント本体の実装コードをロードし，インスタンス化を行う
- 記録してあったメッセージをコンポーネントに転送する（本来，もし本体が存在していればそのメッセージの内容がコンポーネントに反映されているはずであるため）

以上，ステップ 1 ～ 3 の手順により，コンポーネントが遅延ロードされるにもかかわらず，

（1）メッセージの受信をユーザーに通知すること

（2）コンポーネントを開くと，受信済みメッセージを反映して表示すること

を実現した．

5. モバイル・デバイス向けマッシュアップ環境の構築

図 6 に，構築したテレコム・マッシュアップ環境の概要を示す．その中で，筆者らが実装したウィジェット（インターネット・ウィジェット，テレコム・ウィジェット）およびビュー・アダプテーション機構について述べる．

5.1 ウィジェット（Widget）

今回のシステムにおいて，マッシュアップの対象となる基本要素がウィジェットである．

インターネット・ウィジェットは，インターネット上のさまざまなサービス（天気，地図，ニュースなど）を利用するウィジェットである．サービスの形態は，REST，Web サービス，JavaScript™ API などサービスによってさまざまであるが，ウィジェット化することにより，それらの差異を吸収し，統一的に扱えるようになり，マッシュアップが容易になる．

テレコム・ウィジェットはテレコム・サーバーが提供するテレコム・サービスを利用するウィジェットである．具体的には，テレコム・サーバーが Parlay X API として提供するプレゼンス，位置情報，メッセージ配信（SMS, Short Messaging Service），コール制御などのサービスを活用するためのウィジェットである．Parlay X は通常 Web サービスとして提供されるため，それを利用するウィジェットを開発するには，JavaScript から Web サービスにアクセスする必要がある．JavaScript で Web サービスの SOAP プロトコルを扱うことは不可

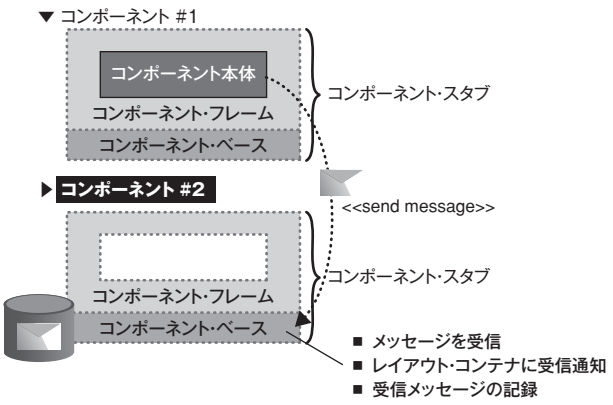

図 4. メッセージの送信

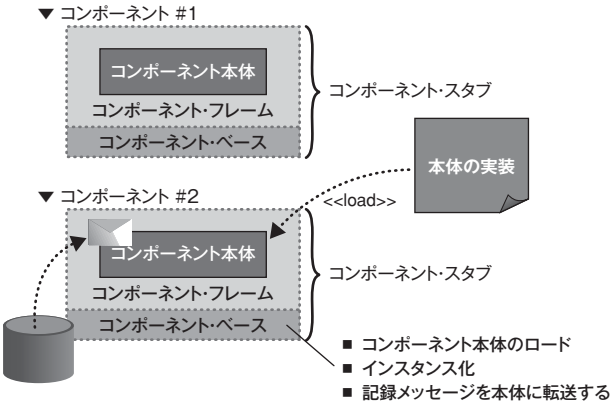

図 5. 記録メッセージの転送

能ではないが，プログラムが複雑になることは避けられない．そこで，図 7 に示すように，Java コードによって Parlay X Web サービスにアクセスし，JSON 形式で結果を返す Servlet を用意した．これにより，クライアントの JavaScript からは REST API 経由でその Servlet にアクセスする構成でテレコム・ウィジェットを実装することが可能となった．

5.2 アセンブリー・ツール（Assembly Tool）

アセンブリー・ツールは，ウィジェットの配置，ウィジェットのプロパティーの設定，ウィジェット間の連携の定義などを行い，目的のマッシュアップ・アプリケーションを組み立てるツールである．もちろん，簡単なプログラムを書けばツールがなくてもウィジェットを組み合わせたマッシュアップ・アプリケーションの作成は可能であり，実際，Web 2.0 が登場した初期のころはそういったスタイルが主流であったが，IT の専門家でないビジネス･ユーザーにとっては敷居が高い．現在は，プログラミングなしに画面上で容易にマッシュアップを行えるツールが各社から提供されている．今回は IBM が提供する IBM Mashup Center を利用した．

5.3 ビュー・アダプテーション（View Adaptation）

IBM Mashup Center は，PC 上でマッシュアップ作業を行うが，作成したアプリケーションも PC 上での使用が前提となっており，そのままではモバイル・デバイスには不向きである．そこで，3.1 節で述べた要件を満たすべく，モバイル・デバイスの狭い画面に適した表示を行い，受信通知や遅延ロードを実現するビュー・アダプテーション機能を実装した．ただし，マッシュアップ作業は PC 上で行うこととし，作成アプリケーションを特定の URL から開くとアイコンや折り畳みセクションのレイアウト・コンテナ上にウィジェットが配置される形で表示される仕組みとした．

ビュー・アダプテーションは，従来技術の Web コンテンツのトランス・コーディングとは異なる．トランス･コーディングは，デバイスの表示能力に合わせてコンテンツを変換する技術であるが,ビュー・アダプテーションは，コンテンツ（=ウィジェット）自体を変換せず，レイアウト・コンテナ上に配置することによってモバイル・デバイス上で扱いやすくするとともに，受信通知や遅延ロードなどの付加機能を与えるものである．

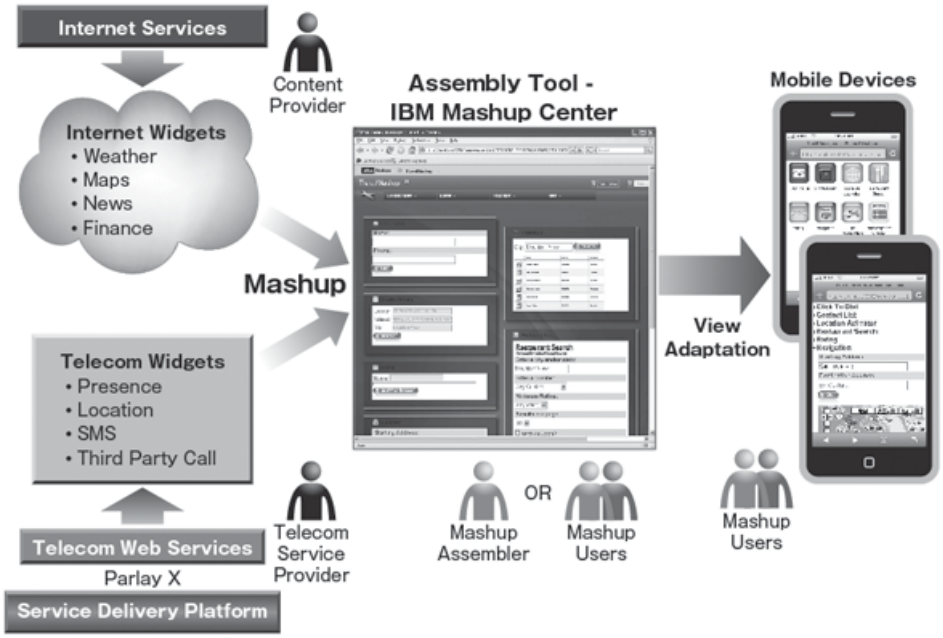

図 6. テレコム・マッシュアップ環境の概要

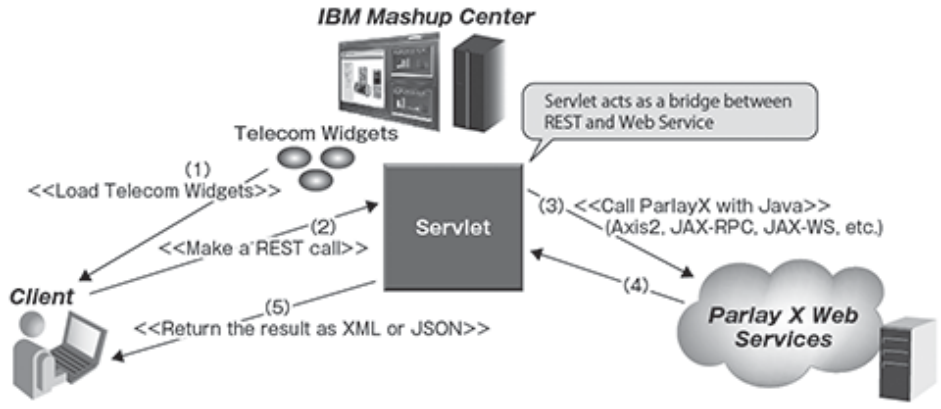

図 7. Web サービス利用のアプローチ

6. 結果と考察

図８は，PC 上で IBM Mashup Center を使ってマッシュアップ作業をしている様子を示す．作業は，ドラッグ＆ドロップによってウィジェットを編集領域に配置し，必要に応じてウィジェット間の連携の設定（ワイヤリング）を行い，ページを保存するだけである．

作成したマッシュアップ・アプリケーションをモバイル・デバイス上で表示した様子を図９に示す．図９左および中央は，iPhone 上でそれぞれセクション・コンテナ，アイコン・コンテナ上にウィジェットを配置した例を示す．図９右は Google Android 上での表示例である．

図 10 は，受信通知の動作例を示す．2 者間の呼を確立する Click To Dial ウィジェットとアドレス帳に相当する Contact List ウィジェットが連携しており，Contact List（図 10 左）のアイテムを選択すると，Click To Dial のセクション･タイトルがハイライト表示（図 10 中央）され，メッセージ受信を知らせる．Click To Dial を開くと，この時点で初めてウィジェットがロードされるが，受信した選択アイテムが反映されていることが分かる．（図 10 右）

以上のようにモバイル・デバイス向けマッシュアップ環境を実現できることを示したが，4 章で提案したコンポーネントを二重構造とするアプローチにはほかにも重要な利点が 2 つ存在する．

1 つは，ウィジェットのマッシュアップ環境への依存を分離できる点である．ウィジェット仕様の標準化の動きもあるが［10］，現状では各社各様であり，例えば IBM Mashup Center では iWidget という独自仕様を採用している．本アプローチによれば，マッシュアップ環境固有のウィジェット仕様への依存はコンポーネント・スタブに封じ込め，コンポーネント本体は Dojo［11］などのオープンな技術のみを利用することで再利用性を高めることができる．

もう 1 つの利点は，コンポーネント・フレーム部分は各コンポーネント固有のイベント定義などを含むが，UI やロジックは一切含まないため，機械的に生成することも難しくない点である．実際に筆者らは，ウィザード形式でパラメーターを指定することによりコンポーネント・スタブやウィジェット本体のひな型を生成する開発支援ツールを Eclipse のプラグインとして実装した．これによりコンポーネントの開発者は本体の UI やビジネス・ロジックの実装に集中できるようになり，その有効性を確認した．

7. おわりに

本論文では，テレコム･サービスなどをウィジェット化し，マッシュアップすることでユーザー独自のアプリケーションを簡単に作成し，モバイル・デバイス上で利用するシステムについて述べた．モバイル・デバイスを対象とする場合，ユーザー・インターフェースの工夫やパフォー

図 8. マッシュアップ・ページの作成

図 9. モバイル・デバイスでの動作例

図 10. 受信通知の動作例

マンスの向上が不可欠であり，それらに対処すべく受信通知と遅延ロードという本来相いれない要件を両立するアプローチを提案した．本手法を使って実際にシステムを構築し，アプリケーションの起動時間はコンポーネントのコード・サイズの影響を受けないことを確認した．さらに，本手法はコンポーネントの再利用性および開発ツールへの親和性の観点からも利点を見いだせることを示した．

また，本システムは純粋な Web ベースであり，モバイル・デバイス上のブラウザーで動作するゼロ・フットプリント（=インストールが不要）のシステムであるが，従来は活用が困難であった通信系のサービスでさえもマッシュアップのような柔軟な形態で利用可能であることを示した．

現在，多くのスマートフォンが開発ツール・キット（SDK）を提供しており，ネイティブ・アプリケーションを開発できるようになっているが，デバイスごとに異なる固有の技術を使わなければならない点やアプリケーション配布の問題などの課題が存在する．これらの課題に対する解の 1 つとして，本システムのような Web ベースのソリューションはますます重要性を増すと考えられる．今後は，テレコム・サービスに限らず，ERP，CRM，SCM などの企業情報システム，大学情報システム，システム・モニタリング，在庫管理，金融機関向け情報系システム，コラボレーション・ツールなどのようなさまざまな分野に適用範囲を広げていくことを検討したい．

参考文献

［1］ Gartner Highlights Key Predictions for IT Organizations and Users in 2010 and Beyond, http://www.gartner.com/it/page.jsp?id=1278413 (2010).
［2］ ITU, “General overview of NGN,” ITU-T recommendation Y.2001 (2004).
［3］ 森田・今中・鎌谷・大羽・谷田：“ITU-T の NGN 標準化動向,” NTT 技術ジャーナル，Vol.19，No.9，pp. 111-113 (2007).
［4］ The Parlay Group, http://www.parlay.org/
［5］ Tim O'Reilly. What is Web 2.0-design patterns and business models for the next generation of software. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, (2005).
［6］ 若尾 , 神山 : “Web 2.0 – Web の最新動向 –,” 電子情報通信学会誌 , Vol. 89, No. 12, pp. 1085-1090 (2006).
［7］ Thierry Pollet: “How the web radically transforms communication networks” , INSERTech; Vol. 392, Article No.: 5, Proceedings of the 2007 Workshop on INnovative SERvice Technologies (2007).
［8］ Jong-Lok Yoon, “Telco 2.0: A New Role and Business Model,” IEEE Communications Magazine, Vol. 45, No.1, pp. 10-12 (2007).
［9］ Blum, N. et.al. .; "Definition of a Web 2.0 Gateway for 3rd Party Service Access to Next Generation Networks", in IFIP, Volume 284; Wireless and Mobile Networking; Zoubir Mammeri; (Boston: Springer), pp. 247-258, ISBN 978-0-387-84838-9 (2008).
［10］ OpenAjax Alliance, http://www.openajax.org/
［11］ The Dojo Toolkit, http://www.dojotoolkit.org/

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
Lotus テクノロジー開発
アドバイザリー・ソフトウェア開発エンジニア

神山 淑朗 Yoshiroh Kamiyama

［プロフィール］
1992 年，日本 IBM 入社．Web オーサリングツール，機械翻訳システムの開発を担当．2003 年以降，リッチクライアント，シチュエーショナルアプリケーション開発ツール，Web メールクライアント，モバイル Web アプリケーションなどの研究開発に従事している．

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
シニア・テクニカル・スタッフ・メンバー

野口 雅人 Masato Noguchi

［プロフィール］
1990 年，日本 IBM 入社．ホームページ・ビルダー，Rational Application Developer などの製品開発に従事．現在は Web 2.0/RIA 関連のソフトウェア開発を担当．