

携帯電話向けモバイル XML データベース用ミドルウェアの開発

小高 健二 並木 美太郎

東京農工大学 工学教育部 情報コミュニケーション工学専攻

1. はじめに

携帯電話への Java 実行環境の搭載により、携帯電話は「プラットフォーム独立でプログラマブルなデータ通信機能付 PDA」として利用可能なほど高機能化した。しかし、携帯電話向け Java 実行環境上のモバイル AP (アプリケーション) はゲームなどエンターテインメント系が中心で、その能力を十分に生かしているとは言い難い。

一方、Web 上のコンテンツ作成において汎用的なデータフォーマットとして XML が注目され、多くのシステムで利用されている。モバイルアプリケーションにおいても XML を利用することで、開発コストの低減、データの汎用性や再利用性の向上などを実現することができる。

そこで、携帯電話向けモバイル XML データベース用ミドルウェア (以下、本システム) では、携帯電話の Java アプリの中にコンポーネントとして組み込むことにより、携帯電話上に Web 上のコンテンツとの連携が可能となる XML をデータ形式とするデータベースを構築し、Web 上のコンテンツを活用した実用的なモバイル AP の作成を支援する。これにより、携帯電話上で XML をデータフォーマットとするクライアント指向の実用的なモバイル AP の作成が可能になる。

2. 目標

本システムの目標は以下の 3 つである。

(1) XML DB の構築を実現するコンポーネント

携帯電話向け Java 実行環境は複数のアプリ間でのデータやプログラムの共有ができないため、本システムを複数の AP で利用できる共有のライブラリという形で提供することはできない。そこで、本システムは AP のパッケージの中にコンポーネントとして組み込むことで、携帯電話上での XML DB の構築を実現する。

(2) 端末上での XML の検索・更新

CPU やメモリ資源の制限された携帯電話向け Java 実行環境で XML を処理することは、大きな負荷となる。しかし、電波の届かない場所でも利用可能なクライアント指向の実用的な AP の実現には端末上で XML に対する処理を行えるようにすることが必要である。

(3) 端末内ストレージへの XML の保存・管理

端末内のストレージに XML をストアし、管理することができなければ、携帯電話の電波が届かない場所では利用できなくなってしまう。Web や端末外のメモリとの連携を行う場合においても、大きなデータベースから一部を切り出して端末内で処理するなどの利用法が考えられるため、必須の機能である。

3. 設計方針

前節の目標を実現するため、以下の設計方針を挙げる。

(1) プロファイル独立な API

本システムは、NTT ドコモ「i アプリ」の DoJa 仕様[1]と、Vodafone「V アプリ」などの J2ME MIDP 仕様[2]の携帯電話をターゲット端末とする。これらは J2ME CLDC 仕様をベースとして携帯端末向けの API を追加したものだが、ストレージ管理や GUI などの API が異なっている。そこで、本システムではデータ管理についてプロファイル依存の部分を吸収し、AP 作成者に特定の端末のプロファイルに依存しない API を提供することで、AP の開発コストを削減する。

(2) コンパクトサイズ、省メモリと性能の両立

現在、Oracle 9i Lite[3]や IBM DB2 Everyplace[4]などといったモバイル DB があるが、これらはメモリ使用量が多く携帯電話向け Java 実行環境への実装が難しい、クライアント指向ではない、などの問題がある。そこで、本システムは必要最低限の機能のみを搭載し、JAR アーカイブで数十 KB 程度、現在でも最新の端末ならば搭載可能なサイズとする。また、携帯電話の高機能化により、Java 実行環境の性能は年々上がっているが、CPU 性能やヒープ領域などは充分とはいえないため、本システムでは API に応じて内部での XML ファイルの扱いをストリームとランダムアクセスで切り替えることによって無駄なメモリの使用をなくし、性能の向上を目指す。

4. システムの構成と機能

4.1. 全体構成

本システムは以下の 3 つのコンポーネントから構成される。

(1) XML DB コンポーネント

(2) XML パーサ

(3) ファイル管理コンポーネント

全体構成を図 1 に示す。

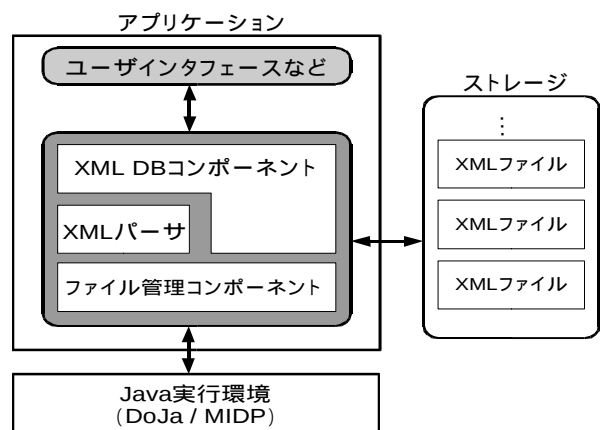


図 1 全体構成

「Development of a Middleware for Mobile XML Databases on Cellular Phones」

Kenji ODAKA, Mitaro NAMIKI

Graduate school of Engineering, Tokyo University of Agriculture and Technology

o-kenji@namikilab.tuat.ac.jp, namiki@cc.tuat.ac.jp

4.2. XML DB コンポーネント

XML DB コンポーネントは AP 作成者に対するインタフェースを提供する部分であり、AP はこのコンポーネントを通してストレージ内にストアされた XML ファイルの操作や、XML パーサを用いた XML データに対する検索・更新を行う。

API については XML:DB Initiative[5]が策定している XML Database API 仕様をベースとしたものとする。本来 XML Database API では検索に XPath、更新には XUpdate を用いるが、どちらも DOM ツリーの構築が必要なものとなっており、携帯電話向け Java 実行環境上での実現は難しい。そこで、本システムでは主に DB 内の XML リソースの操作を行うためのインタフェースとして XML Database API を利用し、検索などについては独自の API を用意するものとする。

本システムで扱う XML は、通常の XML フォーマットから DTD と実体参照を省いた compactXML[6]である。本システムは、妥当性及び整形形式であることについては既に確認してあることを前提とし、メモリ使用量などの増加を招く DTD についてはサポートしないものとする。その結果、DTD を利用して宣言する実体参照についても非対応となる。

4.3. XML パーサ「cSAX」

端末上での XML に対する操作を実現するためには、XML の解析を行う XML パーサが必要となる。本システムでは、XML パーサとして本研究室で開発された cSAX[6]を利用する。cSAX はイベント駆動型の XML 解析インタフェースである SAX を compactXML の解析を行うためにサブセット化した API である。

4.4. ファイル管理コンポーネント「FML」

ファイル管理コンポーネントとして本研究室で開発が行われている FML を用いる。FML は、端末内のストレージへの XML の保存・管理操作や、Web 上のデータベースや外部メモリとの連携を実際に行う部分である。FML は端末内のストレージやネットワーク、外部メモリなどのアクセス方法の違いを吸収し、ファイルを 1 つのデータの単位として管理、J2SE の File クラスに似た API を提供することで、プロファイルごとの違いを吸収する。

5. API の仕様

データベースコンポーネントの API は、XML Database API をベースとしたものとなっているため、基本的なメソッドは XML Database API と同じものとなっている。主なメソッドを表 1 に示す。

6. 実装と評価

現在までに、XML DB コンポーネントの XML などのリソースを扱う基本部分と、cSAX を用いた簡単な検索機能についての実装を行った。

ファイル管理コンポーネントについてはデバッグ中であるため、MIDP をベースに独自のファイルシステム API を実装している Vodafone の JSCL 仕様[7]の端末において簡単なラップを作成し、V602SH を用いて処理時間を測定した。この検索ルーチンは cSAX を用いて XML 全体を順に読みながら処理を行うものとなっているため、実行時間はファイルサイズに比例する。

表 1 XML DB コンポーネントの主な API

<code>getCollection(java.lang.String uri)</code>
URI から一つのコレクション（リソースをまとめるためのディレクトリのようなもの）を得る
<code>createResource(String id, String type)</code>
DB に空のリソース（XML など）を作成する
<code>getResource(String id)</code>
DB から id で指定したリソースを取り出す
<code>getContentAsCSAX(HandlerBase handler)</code>
XML を cSAX ハンドラとして取り出す
<code>setContentAsCSAX()</code>
cSAX ハンドラを使って新しい XML を DB に保存する
<code>getElementCount()</code>
要素数をカウントする
<code>listContent(String tag)</code>
指定したタグと一致するタグの内容を返す

実行結果はファイルサイズ 60KB、要素数 1675 個の XML ファイルを用いた場合で約 13 秒となった。現在の携帯電話用 Java 実行環境においては 100KB 以上の XML ファイルを扱うことも予想されるため、処理速度の向上が必要である。しかし、携帯電話の I/O 性能は PC と比べて大きく劣っており、シーケンシャルに XML 全体を読むような方式で性能を稼ぐことは難しい。改善方法としては、検索インデクスとランダムアクセスを組み合わせることや、パッファを用いることなどが考えられる。

また、性能以外には更新に関する API の検討がまだ不十分であることが問題として挙げられる。

7. おわりに

本論文では、携帯電話向けモバイル XML データベースの核となる部分について、概要を述べた。ミドルウェアの基本的な部分の開発を終えたあとは、性能向上のためのインデクス作成方法、DB 更新のための API 拡張などの検討を行う。ただし、これらはサイズやメモリとのトレードオフとなるため、最新の端末には載せられるサイズを維持する。また、本システムを利用した AP を作成し、本システムの有効性と活用法を示したい。

参考文献

- [1] 「i モード対応 Java コンテンツ開発ガイド ～詳細編～ 第 1.0 版」（株式会社 NTT ドコモ、2000）、「for DoJa 3.5」（株式会社 NTT ドコモ、2004）
- [2] J2ME Mobile Information Device Profile (MIDP) (<http://java.sun.com/products/midp/>)
- [3] Oracle 9i Lite (http://www.oracle.co.jp/o9i_lite/)
- [4] IBM DB2 Everyplace (<http://www-6.ibm.com/jp/software/data/db2/everyplace/>)
- [5] XML:DB Initiative (<http://xmldb-org.sourceforge.net/>)
- [6] 益子由裕、並木美太郎「携帯電話向けの XML 処理用ミドルウェア」情報処理学会論文誌, Vol.45, No.ACS5, pp.79-90
- [7] 「V アプリ開発ガイド [開発編] 1.0.4」（ボーダフォン株式会社、2004）