J2ME 環境での XML パーサ

難波亮丞 <rna@horobi.com>

本報告は J2ME 環境、特に携帯電話の Java アプリケーション実行環境において、軽量 XML パーサ実装が実用的に動作することを実証し、当該環境でのプログラミングにおいて注意すべき点をまとめたものである。

XML Parsers on J2ME Environment

Ryosuke Nanba <rna@horobi.com>

This report proves that light-weight XML parsers work fine on the J2ME of cellular phone devices, and picks up tips for programming on that environment.

この報告について

この報告は J2ME 環境、特に携帯電話の Java アプリケーション実行環境において、XML パーサを使用する際の注意点や問題点について調査したものである。ここで使う XML パーサは通常のバイナリ化されていない XML データを扱うものとする。また、読み込んだ XML データの処理や XML データの生成・出力などについては扱わない。

J2ME 環境の概要

J2ME (Java2 platform, Micro Edition) は組み込み機器、PDA、携帯電話などのような環境に特化した Java 環境の仕様である。日本では携帯電話キャリア各社が J2ME 環境を内蔵した端末を採用したことで、特に携帯電話向けアプリケーションの主要なプラットフォームとして注目されている。以下、各社の携帯電話の J2ME 環境の仕様や制限などについて、XML パーサを動作させる際に注意が必要な点を見ていく。

J2ME と言っても API の互換性や実行環境の制限などの点で各社異なるのが実情である(参照: [iapp], [japp], [ezplus])。その概要を表1にまとめた。これらのうち API の違いについては XML パーサに直接関係しない。どれも CLDC (Connected, Limited Device Configuration) に準拠しており、XML パーサの機能は CLDC の API だけで実装できる。重要なのは、アプリケーション(JARファイル)のサイズとHTTP通信による一回の転送量の制限である。

表1: サービス毎の仕様と制限

サービス名	キャリア	API	JARサイズ上限	HTTP通信上限		接続先制限	XML受信に使用可能な メディアタイプ
				下り	上り		
iアプリ (503i)	NTTドコモ	CLDC 1.0 / DoJa 1.0	10KB	10KB	5KB	ダウンロード先と同一サイト	制限なし
iアプリ (504i)	NTTドコモ	CLDC 1.0 / DoJa 2.0	30KB	10KB	5KB	ダウンロード先と同一サイト	制限なし
Javaアプリ	J-Phone	CLDC 1.0 / MIDP 1.0 / JSCL	30KB/80KB*	80KB	12KB	無制限(認可が必要)	application/java-archive
ezplus	KDDI	CLDC 1.0 / MIDP 1.0 / KDDI Profile	50KB	9000byte	9000byte	3箇所まで設定可能	application/octet-stream

＊ 80KB はパケット通信対応端末の場合。

アプリケーションが XML を利用する場合最低必要となるのが XML パーサだが、JAR ファイルが数10KB程度に制限される携帯電話の環境では使用できるパーサは限られてくる。

また、携帯電話の環境では HTTP 通信の転送量は 10KB 程度にまで制限されている。もっと制限が緩い場合でも現状では通信速度と課金を考慮してそのぐらいのサイズのデータを主に扱うことになるであろう。一般の XML パーサはもっと大きなサイズのデータでテストされているため、一般に高性能と言われているパーサが有利になるとは限らない。

メディアタイプの制限については注意が必要である。メディアタイプの制限はキャリアのゲートウェイが HTML や画像など特定のメディアタイプのコンテンツしか通さないことから生じる制限である。この制限を避けるためには、サーバー側でコンテンツに本来のメディアタイプ(SVG なら image/svg+xml など)ではないメディアタイプ(例えば application/java-archive)を設定するという、一種の偽装を行う必要がある。携帯電話側のアプリケーションでは HTTP レスポンスのメディアタイプを意識することはほとんどないので問題ないが、同じサーバーが一般の PC にも同じデータを配信している場合には、PC 側のブラウザでデータが表示できなくなる(ダウンロードになってしまう)という問題が発生しうる。

また接続先の制限についても知っておきたい。XML データに DTD や外部実体、XML Schema 文書などへの参照がある場合、接続先制限によって読み込めなくなることがある。現在のところ J2ME で使えるパーサでそれらの外部リソースへの参照を行えるものはほとんどないので実際には問題にならないかもしれないが、将来に向けて留意すべき点ではある。

軽量XMLパーサ

Java で動く XML パーサは数多いが、携帯電話の J2ME 環境で動くものは少ない。J2ME のAPIの制限からビルドできない、コードサイズが大きすぎる、メモリ使用量が大きい、などの問題で動作しない場合が多い。

そのような問題がないか、比較的簡単な作業でクリアできるものを表2に挙げた。コードサイズについてはパーサが動作する最小の構成で JAR ファイルのサイズが 20KB 程度以下になるものを選んだ。ビルドについては主観的ではあるが、ビルド時のエラーが 100 個以上出るものについてはその内容を検討し修正困難と判断したものを除外した。エラーが 100 個以内のものについては動作可能な状態になるまでソースコードを修正してテストした。

とりあえず動作はしても、実際に性能的・機能的な点で実用に耐えるかどうかという問題はあるが、この点については次章以降で検討する。また、ビルド時に必要な修正作業の詳細については筆者の Web サイトで公開している資料［src］を参照されたい。

なお、ここに挙げたものはいずれもソースコードを含め無料で入手できる。ただし再配布についてはライセンス上制限される場合がある。Min は独自のライセンス、それ以外はオープンソースライセンスであるが、BSD スタイルのライセンスでも内容が微妙に異なるものがある。詳しくは各パーサの付属文書等を参照されたい。

表2: 各種軽量パーサ

パーサ名	バージョン*	API	ライセンス	入手先
kXML2	2.1.6	Pull + 独自tree	CPL	http://www.kxml.org/
MXP1	1.1.2	Pull	BSDスタイル	http://www.extreme.indiana.edu/xgws/xsoap/xpp/mxp1/
MinML1	1.7	SAX1	BSDスタイル	http://www.wilson.co.uk/xml/minml.htm
MinML2	0.3	SAX2	BSDスタイル	http://www.wilson.co.uk/xml/minml2.htm
TinyXML	0.7	独自(event + tree)	GPL	http://www.gibaradunn.srac.org/tiny/
NanoXML/Lite	2.2.2	独自(tree)	zlib/libpng	http://web.wanadoo.be/cyberelf/nanoxml/index.html
Min	1.05A	SAX1	独自	http://www.docuverse.com/min/

＊ バージョンは2002年12月15日時点での最新バージョン。

API のうちパーサ独自のものについては「各種パーサの特徴」のパーサ毎の説明を参照されたい。

互換性

XML 1.0 への準拠性

現在のところ XML 1.0 に完全に準拠した軽量パーサはほとんどない。多くのパーサは何らかの制限付きのサブセット仕様を満たすものになっており、妥当性検証はもちろんのこと XML 1.0 で定められた non-validating パーサの仕様にも厳密には従っていない。テスト対象とした各パーサの XML 1.0 の仕様に対する対応状況を 表3にまとめた。

表3: XML各機能との互換性

パーサ名	混在内容	空要素	CDATA セクション	コメント	属性	文字参照	定義済実体	内部実体	外部実体	デフォルト属性
kXML2	○	○	○	○	○	○	○	△*1	×	×
MXP1 (min)	○	○	○	○	○	○	○	△*1	×	×
MXP1 (std)	○	○	○	○	○	○	○	○	×	○
MinML1	△*2	○	○	○	○	○	○	×	×	×
MinML2	△*2	○	○	○	○	○	○	×	×	×
TinyXML	○*3	○	○	○	○	○	○	○	△*4	△*5
NanoXML/Lite	×	○	○	○	○	○	○	×	×	×
Min	×	×	×	×	×	○	×	×	×	×

*1 アプリケーション側で実体を定義するコードが必要。

*2 原則として空白のみのテキストノードを捨てる。

*3 バグフィックスパッチを適用した場合。元々はタグの後ろに続く空白を無条件に捨てるというバグがある。

*4 アプリケーション側で外部実体を解決するコードが必要。

*5 デフォルト値は自動的には適用されないが、ATTLIST のイベントが来るので自力で実装すれば可能。

まず、ほとんどの軽量パーサは DTD の内部サブセットを一切解釈しない。non-validating パーサは妥当性検証はしないが、DTD の内部サブセットがあればそれをパースして内部実体の置換とデフォルト属性の設定を行うことが要求される。

軽量パーサにとってこれは負担になるので XML Pull API などではこの機能をオプションとしている。内部サブセットのある文書を扱うことは比較的まれであるし、デフォルト属性の効果は元々 DTD を参照しない XML データを扱うためにアプリケーション側でハードコードされることが多いので、実際にはこの制限はあまり障害にはならない。よって以下ではこの種のパーサを準フルセットパーサと呼ぶことにする。

文書に対してある種の機能を使わないことを要求するパーサもある。以下この種のパーサをサブセットパーサと呼ぶ。サブセットパーサの要求には、例えば混在内容(要素内容で、文字列と子要素が混ざったもの)を許さないというものがある。混在内容は XHTML のような文書指向の文書フォーマットでは多用されるが、データ指向の文書フォーマットでは全く使わない場合も多い。例えば住所録フォーマットの ContactXML [cxml] などがそれに該当する。

この種の制限で最も厳しいものは文書が XML のサブセット仕様である Minimal XML に準拠することを求めるものである。Minimal XML は基本的に開始タグと終了タグと文字参照しか扱わないため、これを要求するパーサでは通常の XML データはまず扱えない。通常の XML データを 機械的に Minimal XML に変換することは可能であるが、一般には、専用の DTD と運用規則(コメントさえも使えないので)の元で作成されるデータのみを扱うことになる。このような厳しい制限と引き替えにコンパクトで高速なパーサの実装が得られる。

エンコーディング

J2ME 環境では通常の Java 環境と異なり、利用できるエンコーディングに制限がある。特に日本の携帯電話の場合、基本的にはシフトJIS (SJIS) と ASCII (US-ASCII) しかサポートしていない(iアプリの場合は SJIS のみ)。XML パーサも一部を除いて環境のサポートするエンコーディングしか扱えない。また軽量パーサの場合は XML の encoding 宣言に記述される IANA のエンコーディング名を適切な Java のエンコーディング名に変換するようなこともしない。

よって、通常はパース前にあらかじめ SJIS で Reader を作成しておきそれをパーサに渡してパースするという手順が望ましい。それでも敢えて InputStream を使用してパーサのエンコーディング検出機能を使う場合、どこまであてにできるかを表4にまとめた。

表4: InputStream 使用時のエンコーディング対応状況

パーサ名	UTF-16/UTF-8 自動検出			encoding宣言の扱い	デフォルトの動作
	検出の可否	UTF-16 読み込み	UTF-8 読み込み		
kXML2	○	環境依存	環境依存	InputStreamReader のコンストラクタにそのまま渡す	UTF-8 を使用
MXP1	×	-	-	一切扱わない	エラー
MinML1	×	-	-	一切扱わない	環境のデフォルトエンコーディングを使用
MinML2	×	-	-	一切扱わない	環境のデフォルトエンコーディングを使用
TinyXML	○	○	○	ASCII, UTF-8, UTF-16系以外は検出された時点でエラー	UTF-8 を使用
NanoXML/Lite	-	-	-	一切扱わない	Reader からのみパースする
Min	○	○	○	XML宣言があるとエラー	UTF-8 を使用

表から分かるように一般のパーサと違い、軽量パーサでは encoding 宣言自体を扱わない場合が多い。今回テストしたパーサの中では kXML2 のみが encoding 宣言を扱うが、宣言された値をそのまま InputStreamReader のコンストラクタに渡すため、SJIS (これは IANA 名ではない)しかサポートしない環境では事実上使えない。

なお一部のパーサ(例えば TinyXML と Min)は自前で UTF-16 系や UTF-8 の Reader を実装しているため、これらをサポートしない J2ME 環境でも UTF-16 や UTF-8 の XML データをパースできる。

ベンチマーク

テスト対象のパーサについて携帯電話実機の J2ME 環境で簡単なベンチマークテストを実施した。テスト内容は 10KB 程度の XML データをパースし、要素と属性と文字の数をカウントするというものである。

テストの手順は以下の通りである。

- ベンチマークアプリケーションを起動
- テストデータに対して6回のテストを実施しそれぞれの処理時間を測定。
- 2回目以降の5回の測定値の平均をパース時間とする。
- 初回の測定値からパース時間を引いた値を起動時間とする。
- カウントが正しい値になっているか確認する。
- 以上を3回繰り返し、起動時間とパース時間の平均値をとる。

テストデータはベンチマークアプリケーションの JAR ファイル内のリソースからあらかじめ適切なエンコーディングを設定して作成した InputStreamReader を介して読み込んだ。ただし TinyXML については Reader からパースができないので InputStream から読み込んだ。

テストデータは三種類用意した。その内容を表5にまとめた。

表5: テストファイル

ファイル名	サイズ	文書型	符号化方式	要素数/属性数/文字数	マークアップ密度
svg1	10143	SVG 1.0	us-ascii	93/473/781	92%
xhtml1	10223	XHTML 1.0	us-ascii	130/53/7665	26%
xhtml2	9626	XHTML 1.0	Shift_JIS	130/53/4293	- *

* マルチバイトエンコーディングのファイルであり、他のファイルとは単純に比較できないので値は省略した。

表中の「マークアップ密度」は XML データの中でタグが占める割合を表す大まかな指標である。パーサの性能評価の場合、 1 - 文字内容のサイズ / ファイルサイズ で表される。一般にマークアップ密度の高いデータ程パーサに負荷がかかる。

なお、Minimal XML 準拠の Min に対しては Min に付属する minimizer というツール(を非ASCII文字の出力に対応させたもの)でテストファイルを Minimal XML に変換したものを使用した。

ベンチマークアプリケーションは DoJa 2.0 に準拠したもので、iアプリとして動作するものである。各パーサのクラスはパースに必要な最小限の構成でパッケージングした。詳細はベンチマークアプリケーションのソースコード[src]を参照されたい。なお [src] では CLDC/MIDP 版のベンチマークアプリケーションのソースコードも公開している。

iアプリ対応携帯電話である P504i 上で実施したテスト結果が表6である。各パーサのコードサイズ(パーサ実装のJARファイル換算のサイズ)も合わせて示した。kXML2 と MXP1 で (ns) とあるのは名前空間サポートを有効にした場合の結果である。

注意して頂きたいのはベンチマーク結果の絶対値はもちろん、パーサ毎の傾向も端末機種によってかなり変動する可能性があるという点だ。端末が使用している VM の実装、リアルタイム OS、CPU、メモリ環境等々の違いは測定値に大きな影響を与える。そのため、ここでのパース速度や起動速度の順位は絶対ではない。機種が変われば、あるいはファームウェアのバージョンが違うだけでも変動しうると考えて欲しい。*

* 実際、同じベンチマークを J-Phone 端末 J-T51 で実施したところ今回の結果とはかなり異なる傾向が出た。

表6: ベンチマーク結果 (P504i使用)

パーサ名	コードサイズ*1	パース時間 (ms) svg1/xhtml1/xhtml2	起動時間*2 (ms)	結果
kXML2	11.4KB	2639/2416/1907	343	○
kXML2 (ns)		2852/2527/1974	439	○
MXP1(min)	17.9KB	1813/1390/1272	480	○
MXP1(min)(ns)		1648/1336/1240	380	○
MinML1	13.6KB	2467/2280/1837	405	△*3
MinML2	16.1KB	2630/2349/2032	542	△*3
TinyXML	9.8KB	2726/2340/-*4	577	○
NanoXML/Lite	6.8KB	5103/-/-	1084	△*3
Min	14.8KB	1307/970/908 *5	326	-

*1 パーサを一緒にパッケージしたベンチマークアプリケーションのJARファイルサイズからパーサを含まないアプリケーションのJARファイルサイズを引いた値。

*2 初回起動時のパース時間から2回目以降のパース時間を引いた値。

*3 要素間の空白の数だけ文字数が少なくカウントされた。

*4 Shift_JIS に非対応なため xhtml2 はエラーになった。

*5 Minimizer で Minimal XML 化したテストファイルを使用。

各パーサの特徴と評価

表2 で挙げた各パーサの特徴と、機能および性能の評価を以下に示す。繰り返すが性能に関してはあくまで特定の機種でのベンチマーク結果を元にした評価である。機種が違えば評価も変わりうるという点に留意されたい。

パース速度に関しては MXP1 と Min が特に速いことと、NanoXML/Lite が特に遅いことを除けば大きな差は見られなかった。起動速度も NanoXML/Lite が遅いのを除けば大差はない。パーサ選択の際にはコードサイズ、API、将来性などがより重要な要因になると考えられる。

kXML2

Stefan Haustein 氏が開発した XML Pull API 準拠の準フルセットパーサ。オープンソースアプリケーションサーバ Enhydra のコミュニティが開発した kXML を元に軽量化を徹底し、J2ME 準拠の実装にしたものである。ツリー型 API 実装の kdom も含まれているが今回は評価していない。

最初から J2ME 準拠の実装で一切修正なしでビルドできるのは他にはない特徴である。パース速度は平均的だが、起動速度は最も速い部類に入る。名前空間サポートを有効にするとパース速度が 5% 程低下する。起動速度も 3 割弱低下するが、それでも平均的な速度である。コードサイズは準フルセットパーサの中では最小の部類に入る。

比較的バランスのとれた実装で携帯電話の環境に適している。改良やメンテナンスも頻繁に行われており今後についても期待できる。

MXP1

Aleksander Slominski 氏が開発した XML Pull API 準拠のパーサ。準フルセット版(min)とフルセット版(std)があるが、後者は今回は評価していない。コンパクトで高速なパーサとして知られており、現在のところ世界最速と言われている。J2ME 環境よりも J2SE/EE 環境で高速に動くことを重視しているようで、そのままでは J2ME でビルドできない。テストに際してはわずかではあるが修正が必要だった。

パース速度は Min を例外とすると最も高速であった。kXML2 の 2 倍近い速度で体感可能なレベルの違いがある。起動速度も悪くない。一般的には名前空間サポートを有効にすると速度が低下するものだが、MXP1 は逆に若干速くなった。

コードサイズは今回テストした中では最大であった。17.9KB は十分小さいとも言えるが、30KB が上限の iアプリ環境では厳しい。

高速性は魅力的だがコードサイズが難点。改良やメンテナンスは頻繁に行われており今後については期待できるかもしれない。

MinML1,2

John Wilson 氏が開発した組み込み用パーサで MinML1 が SAX1、MinML2 が SAX2 に準拠している。そのままでは J2ME でビルドが通らないが、J2ME 環境では使われない一部の機能を削除することでビルド可能であった。

SAX API を使用したい場合は唯一の選択肢となる。ただし、元々 XML-RPC のような用途を想定しており、最近のバージョンまで混在内容を扱えなかった。その名残で、今のバージョンでも原則として要素並びの隙間にある空白のみの文字データは捨ててしまう。

MinML1 のパース速度は平均的。起動速度は悪くない。コードサイズが若干大きい。MinML2 は SAX2 API となり名前空間サポートが有効になっている(無効にはできない)が、パース速度は MinML1 より 5% 程低下する。起動速度は 3 割以上低下し、やや遅い部類に入ってしまう。

メンテナンスは続いているがあまり活発ではない。現時点の最終更新は2001年11月である。

TinyXML

Tom Gibara 氏が開発したフルセットの non-validation パーサ。API は独自のイベント駆動型 API とツリー型 API があるが、後者は今回は評価していない。J2ME でビルドするためには多少修正が必要である。

パース速度は平均的だが起動速度はやや遅い部類に入る。フルセットパーサにも関わらずコードサイズは非常に小さく、サブセットパーサの NanoXML/Lite を除くと最小のパーサである。DTD をパースできる唯一の軽量パーサという点でもユニークな存在である。ただし DTD のパース機能を携帯電話用のアプリケーションで生かせる機会は稀であろう。

もっともフルセットパーサと言ってもその完成度には疑問が残る。空白の扱いにバグがある(これは修正した上でテストした)、Reader からパースできないので Shift_JIS のデータを扱えない、外部実体参照を解決するための API に不備があり相対パスの解決に失敗するなどの問題を抱えている(他にもあるかもしれない)。また、ATTLIST をパースしてもデフォルト属性がパース結果にストレートに反映されないのは厳密には仕様違反であろう。

そのような状態でありながら、2000年1月にバージョン0.7がリリースされて以降一切メンテナンスされていない。コードサイズは魅力的だが、自力でメンテナンスする気のある人以外にはお奨めしにくいパーサである。

NanoXML/Lite

Marc De Scheemaecker 氏が開発したサブセットパーサ。混在内容を扱えず、非空白文字の並びの中に開始タグがあると即エラーになる。API は独自のツリー構築型。J2ME でのビルドは多少修正が必要である。

SAX API をサポートしたフルセットの NanoXML/Java もあり、こちらも比較的軽量なパーサだが、J2ME でのビルドが困難だったため今回はテストしなかった。

パース速度は最も遅く、平均的なパーサの 2 倍程遅い。これはツリー構築型のパーサであるため、メモリ確保のオーバーヘッドがあること、要素と属性のカウントをツリー構築後にツリーをトラバースして行わなければならないことなどが原因と考えられる。起動速度も最も遅く、平均的なパーサの 2 倍程遅い。

しかしコードサイズは極めて小さく 7KB を切っている。パフォーマンス上の難点はあるが、唯一 503i シリーズで使える(相当に厳しいとは思うが) XML パーサという点では価値がある。

現在、次のメジャーバージョンアップ(NanoXML3)に向けての作業が進んでおり、今後が期待される。

Min

Docuverse 社の開発した Minimal XML 準拠のサブセットパーサ。API は SAX1。「XML 1.0 への準拠性」で述べたように極めて限定された機能を持つかわりに高速軽量な実装になっている。J2ME でのビルドには多少修正が必要である。

パース速度は MXP1 を抜いて最速であった。起動速度は最も速かったが、値自体は平均的。コードサイズは MXP1 よりは小さいがやや大きい部類に入る。

最速と言っても MXP1 との差は 3 割弱程度である。体感可能な差ではあるが、Minimal XML の非常に強い制限と引き替えにするには微妙なアドバンテージであろう。

低機能なだけに現状で完成形なのかもしれないが、2000年4月の 1.0A5 リリース以降一切メンテナンスされていない。

結論

今回の調査結果から J2ME 環境において XML パーサは実用的に使用可能であると考えられる。

パース速度は一般的なシフトJIS文書で 40Kbps 程度であり、現在の携帯電話の通信速度を考えるとボトルネックにはならない。コードサイズについてはいわゆる第2世代の Java アプリケーションサービスではさほど障害にならない。残念ながら NTT ドコモの 503i シリーズでの使用は厳しいが、NanoXML/Lite なら使える可能性がある。

XML パーサの選択についても、現時点で十分な選択肢がある。今のところ性能とサイズのバランスから kXML2 が最も優れていると思われるが、より高速な MXP1、より小さい TinyXML が選べる。さらに小さい NanoXML/Lite も機能的、性能的な欠点はあるが、条件次第では有力な選択肢になりうる。

参考文献

[iapp]
NTTドコモ i-appli 技術資料 http://www.nttdocomo.co.jp/mc-user/i/java/

[japp]
J-Phone Java アプリ 技術資料 http://www.dp.j-phone.com/java/index.html

[ezplus]
AU ezplus 技術資料 http://www.au.kddi.com/ezfactory/tec/spec/ezplus.html

[cxml]
ContactXML http://www.contactxml.org/

[src]
この報告に関連したソースコード等 http://www.horobi.com/xml/mobile/res/

謝辞

この報告は 2002年11月28日に開催された 2002 XML Japan モバイル XML Day のチュートリアル「モバイル・組み込み環境でのXML - 軽量XMLパーサとバイナリXML」のために筆者らが行った調査に基づくものである。

村田真氏はこの調査の発案者である。貴重な機会を作って頂いた上、予備実験の際には実機上でのベンチマークの実施までして頂いた。橋本賢一氏からは携帯電話の Java 環境でのプログラミングについて様々なアドバイスを頂いた。同環境での経験が乏しい筆者の疑問にも懇切丁寧に答えて頂いた。

お二人と、調査のスポンサーである株式会社ネットラーニングにこの場を借りて感謝したい。