JavaOne℠

Sun's 2002 Worldwide Java Developer Conference℠

Parsing XML in the Java™ 2 Platform, Micro Edition (J2ME™)

XML in a MIDP Environment

Jonathan Knudsen
Technical Writer
Sun Microsystems, Inc.

Overall Presentation Goal

Learn how to parse XML documents in a MIDP environment.

Learning Objectives

As a result of this presentation, you will:

- Understand architectural issues involved in selecting XML as a data transport
- Learn about the current palette of small, open source parsers
- Understand the issues of making an application that is small and performs well on a MIDP device

Speaker's Qualifications

Jonathan Knudsen is a Technical Writer for Sun Microsystems' Wireless Developer site

http://wireless.java.sun.com/

http://jonathanknudsen.com/

Books: Wireless Java™ (Apress), Learning Java™ (O'Reilly), Java 2D Graphics (O'Reilly), Java™ Cryptography (O'Reilly)

Articles: JavaWorld™, O'Reilly Network, EXE

Speaking: SD West 2001, O'Reilly Java™ Conference 2001 & 2000, ICJD 2001

Presentation Agenda

Multi-tier System Architecture

Parser Roundup

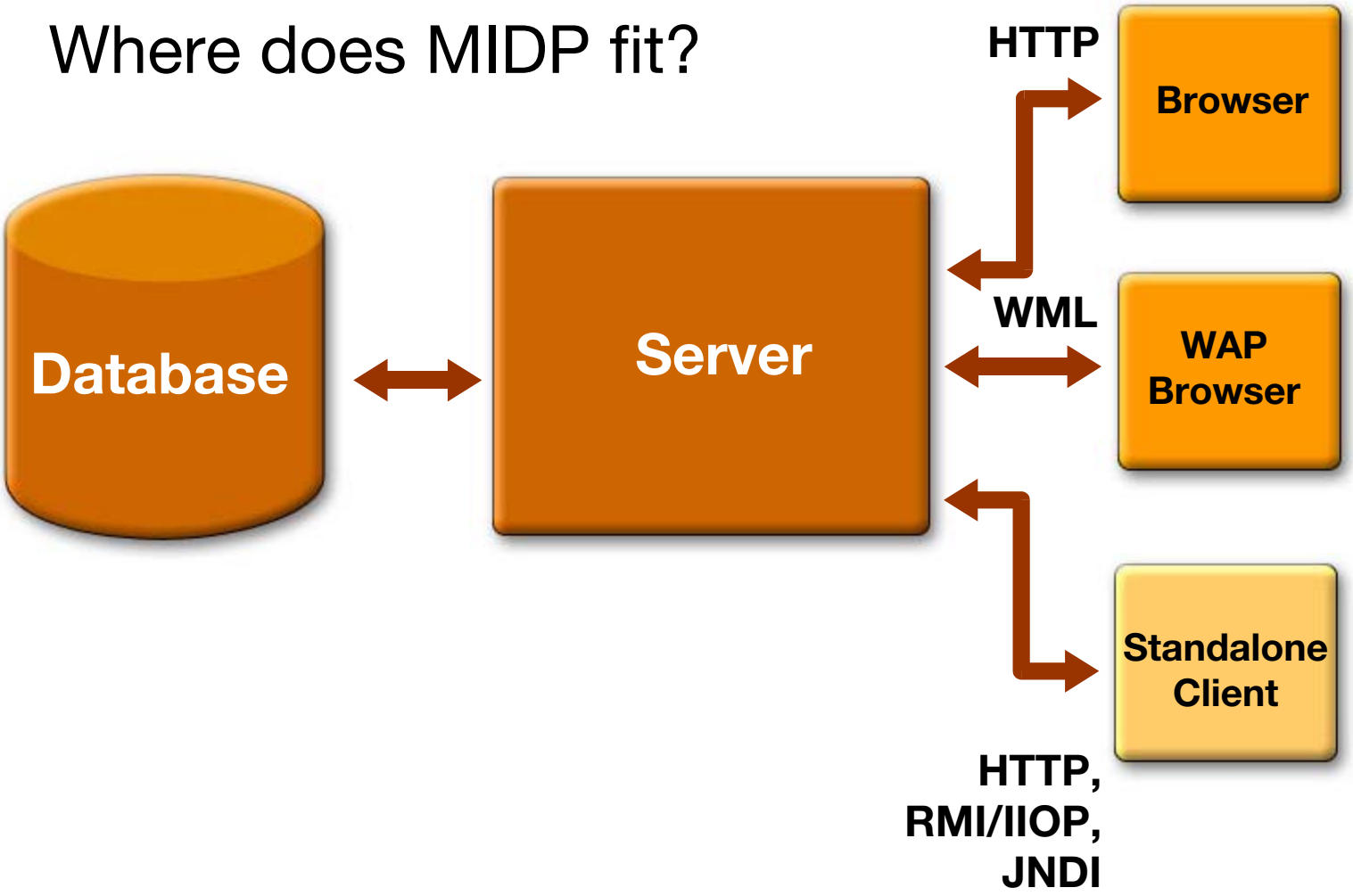
Performance Considerations

Source Code Demonstration

Multi-tier System Architecture

Three Tiers, This Month

Where does MIDP fit?



Everything Is Small in MIDP

Network setup is slow

Data rates are slow

Processor is slow

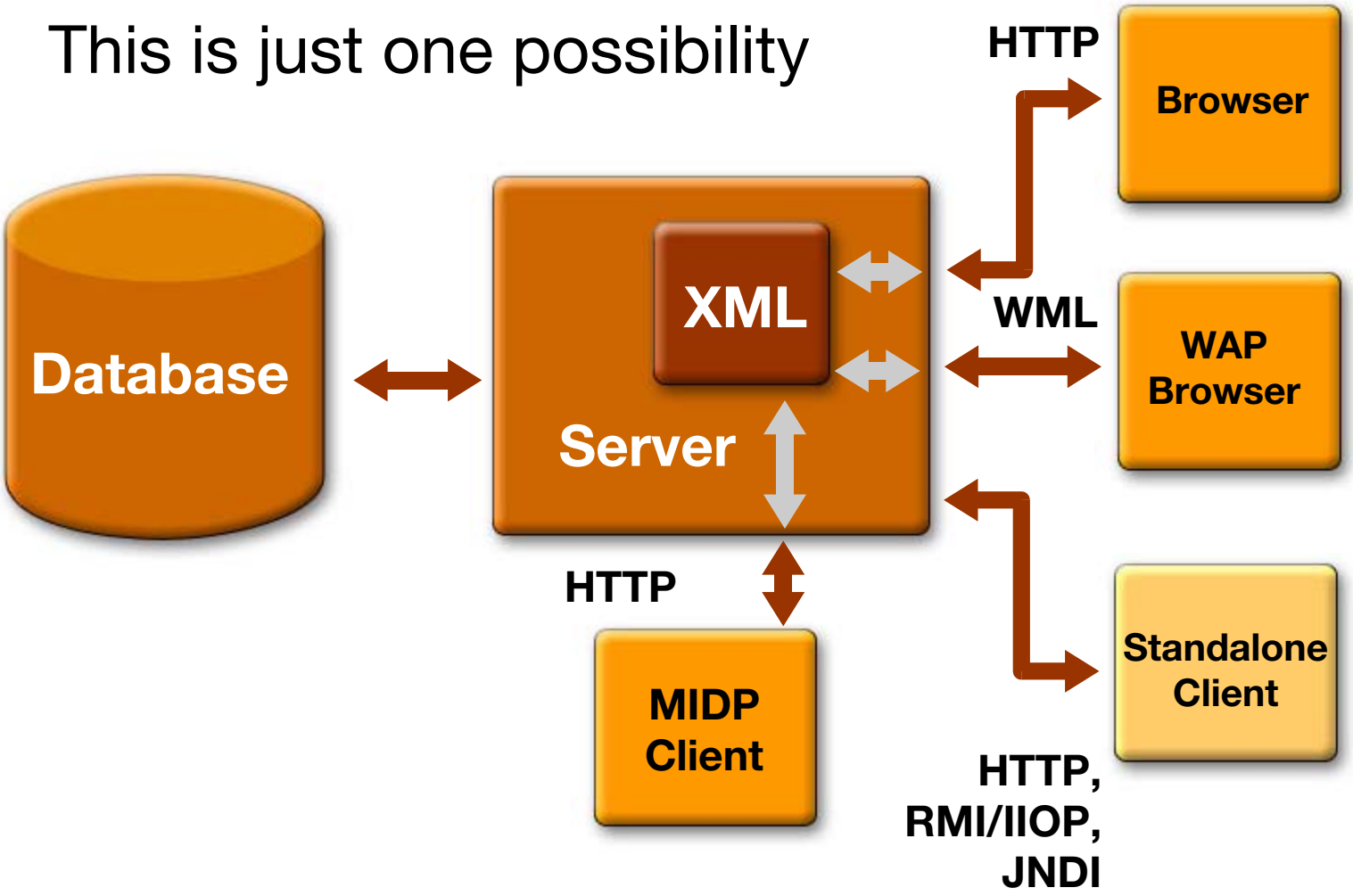
Memory is scarce

MIDP Clients Need Server Support

- No HTML browsers here
- No complex protocols: no JNDI, no RMI
- Server steps up to the plate
  - Mashes data into formats the client understands
  - Handles complex protocols for the client

Three Tiers With MIDP

This is just one possibility



Parser Roundup

Don’t Supersize Me

- Code size is constrained
  - JAR size maximum is about 50 kB (varies by carrier, manufacturer)
- Available memory is generally small
- Open Source is attractive
  - Customizable in size and features
  - Fixable

Parser Types

Model

- Creates an object representation of a document in memory (e.g., DOM)

Push

- Parses through an entire document, spitting out events to registered listeners (e.g., SAX)

Pull

- Parses a little at a time, returning a single element or tag

The Small Parser Lineup

Name	License	Size	MIDP	Type
ASXMLP 011230	**~BSD**	**6 kB**	**yes**	**push**
kXML 2.0 alpha	EPL	9 kB	yes	pull
kXML 1.2	**EPL**	**12 kB**	**yes**	**pull**
MinML 1.7	**BSD**	**14 kB**	**no**	**push**
NanoXML 1.6.4	zlib/libpng	10 kB	patch	model
TinyXML 0.7	**GPL**	**12 kB**	**no**	**model**
Xparse-J 1.1	**GPL**	**6 kB**	**yes**	**model**

Links

Parser	URL
ASXMLP	http://www.alsutton.com/software/xmlparser/
kXML	http://kxml.enhydra.org/
MinML	http://www.wilson.co.uk/xml/minml.htm
NanoXML	http://nanoxml.sourceforge.net/
TinyXML	http://www.gibaradunn.srac.org/tiny/
Xparse-J	http://www.webreference.com/XML/tools/xparse-j.html

Parser	License URL
ASXMLP	http://www.alsutton.com/software/licence.html
kXML	http://kxml.enhydra.org/software/license/
MinML	http://www.opensource.org/licenses/bsd-license.html
NanoXML	http://www.opensource.org/licenses/zlib-license.html
TinyXML	http://www.gibaradunn.srac.org/tiny/gpl.txt
Xparse-J	http://www.webreference.com/xml/tools/license.html

Near Misses

NanoXML 2.2 Lite

- 6 kB
- http://nanoxml.sourceforge.net/

XMLtp 1.7

- 25 kB
- http://members.tripod.de/xmltp/

Porting Techniques

- Remove features you don't need
- Supply missing classes
  - `java.*` naming is questionable
  - Dummy classes or real implementations
- Rewrite unavailable functionality

Performance Considerations

Overview

- Not specific to XML applications
  - An XML parser may push you to the wall
- Runtime performance
  - Connection setup: number of documents
  - Connection speed: document size
- User perception
- Deployment
  - Code size

Document Design

- Connection setup time is long
  - Make each document count
  - Perhaps aggregate documents on the server
- Connection speed is slow
  - Only send essential information
  - Make documents as short as possible

Threading

- Network activity has to go in a separate thread
  - Don’t lock up the application’s interface
  - Ideally, allow the user to do other work while network activity occurs in the background
- Parsing should likely occur in a separate thread
  - Depends on your parser
  - Depends on your document

Code Size

- Carriers or devices may impose restrictions on code size
  - Nextel/Motorola: 50 kB (soft)
  - Devices don’t have much storage space
  - Wireless bandwidth is small
- Code size refers to the size of the MIDlet suite JAR
  - .class files
  - Resource files (images, icons, others)
- Use an obfuscator to reduce class file size

What Does an Obfuscator Do?

Depends on the product: read the documentation

Original purpose was to render code impervious to decompilation

Some possibilities:

- Removes unused classes
- Removes unused methods and variables
- Renames classes, packages and variables
- Adds illegal stuff that confuses decompilers

Using an Obfuscator

The obfuscator may not play nicely with your development environment

Build without obfuscation:

Compile ▸ Preverify ▸ JAR

Build with obfuscation:

Compile ▸ Obfuscate ▸ Preverify ▸ JAR

Another possibility:

Compile ▸ Preverify ▸ JAR4Obfuscate ▸ Preverify ▸ JAR

Some Free Stuff

JAX

http://www.alphaworks.ibm.com/tech/JAX/

Removes unused classes and interfaces

Prunes unused methods and variables

Shortens internal method and field names

Retroguard

http://www.retrologic.com/retroguard-main.html

Renames class, method, and field names

Demo

Summary

- Three-tier application architecture
  - May include XML and transformations for different client types
  - May make sense to send XML to a MIDP client
- Various small parsers exist
  - Differentiated by execution model, license, size
- Optimizations
  - Document size, network connections
  - Code size

If You Only Remember One Thing…

All of this is online.

http://wireless.java.sun.com/midp/articles/parsingxml/

JavaOne℠

Sun's 2002 Worldwide Java Developer Conference™

BEYOND BOUNDARIES

Session 2716