Re-architecting a Digital Library System: Lessons Learned.

University of Michigan Digital Library Production Service:

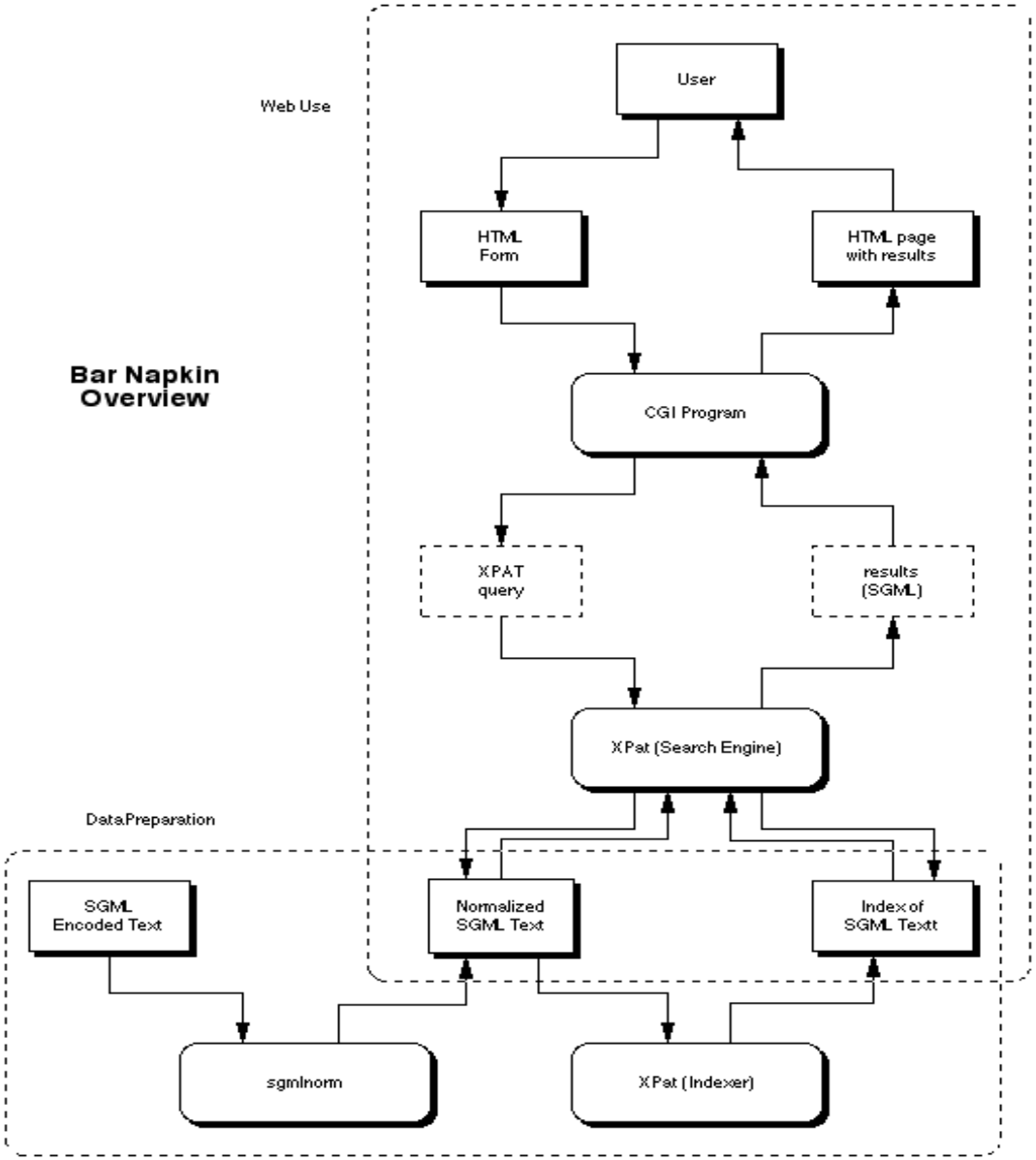
•Phil Farber
•Alan Pagliere
•Chris Powell
•John Weise
•Perry Willett

Outline

- History
- Goals
- Reasons to change
- Data conversion
  - Text
  - Images
- Software
  - XPAT/Unicode
  - Middleware
- Project Management
- Surprises / lessons learned

History

- SSP code (1996)
  - SGML-to-HTML
  - Single perl script
  - One script per collection
  - No cross-collection searching
- DLXS (2000/01)
  - Object oriented design
  - Shared libraries
  - Collection information stored in MySQL db
  - Templates with PIs
  - Fallback


Web Use
User
HTML Form
HTML page with results
Bar Napkin Overview
CGI Program
XPAT query
results (SGML)
XPat (Search Engine)
DataPreparation
SGML Encoded Text
Normalized SGML Text
Index of SGML Textt
sgmlnorm
XPat (Indexer)

Goals

- In addition to adding XML/XSLT/Unicode functionality, what we set out to do:
  - Provide same functionality and services
  - Keep U of M Digital Library operating and updated during development
  - Ease transition, both for ourselves and for other DLXS customers

What we didn’t set out to do

- Create a web-service model
  - No SRU, OpenURL, RSS, Podcast, cell phone…
- Completely rewrite software from ground up
- Change search engines
- Redesign underlying repository

Reasons to Change

- Take advantage of XML and XSLT:
  - Stay current with data formats
  - Simpler to use in a web environment
- Take advantage of Unicode:
  - Unicode supports all world alphabets
  - The UTF-8 encoding is most widely used
- Move formatting and interface issues out of perl middleware:
  - No longer requires a perl programmer to change html output

Data conversion: Text

From SGML to UTF-8 XML

- Conversion of licensed material from vendors (Chadwyck-Healey, Intelex, et al)
- Conversion of locally created material
- Modification of processes for local text creation

A three-step approach

- Convert ISO Latin1 characters to UTF-8
- Convert character entities and numeric character references to UTF-8
- Convert SGML to XML

- From Latin1 é to UTF-8 é
- From é to UTF-8 é
- From é to UTF-8 é
- From é to UTF-8 é
- From <PB N="25"> to <PB N="25"/>

Challenges we faced

- Idiosyncratic entities that needed to be identified in vendor collections
- Some entities had no real Unicode version
- XML and Unicode are not as widely supported in tools as one might think after 10 years as the next big thing
- All collections needed to be completed simultaneously

Tools we used

- For checking UTF-8 validity, jHove and utf8chars
- For converting Latin1 to UTF-8, iconv
- For converting entities to UTF-8, a suite of locally-created tools
- For converting SGML to XML, osx
- As terminal, PuTTY

jHove – what is it?

- The JSTOR/Harvard Object Validation Environment
- Includes a UTF-8 module
- Reports whether your document is or is not valid UTF-8, and which Unicode blocks are contained
- Available at http://hul.harvard.edu/jhove/

iconv – what is it?

- Unix utility program
- Converts files from one encoding to another

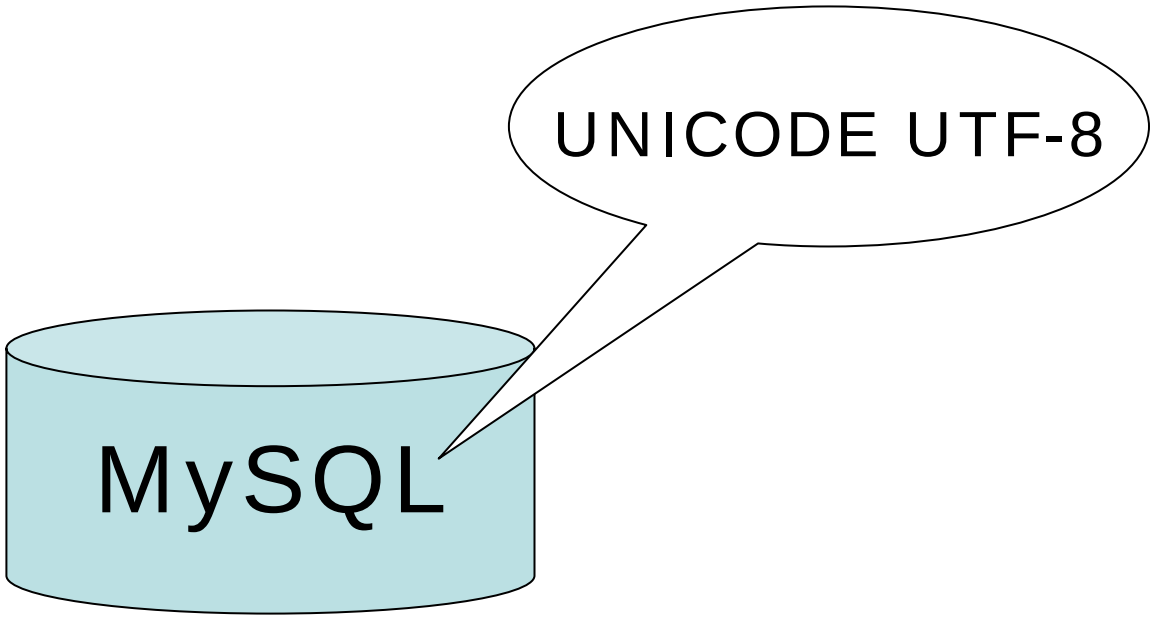
Our locally created tools

- findentities.pl
- utf8chars
- isocer2utf8
- ncr2utf8
- Available as part of the DLXS distribution at www.dlxs.org

osx – what is it?

- Based on James Clark's sx
- Part of Open SP
- Converts SGML documents to XML
- Available at http://openjade.sourceforge.net/

Data Conversion: Images



Anticipated Benefits

- Improved searching.
  - Chichén Itzá = Chichen Itza
- Better browser display.
- XML compliance.

Castillo
Toltec-Maya
Chichén Itzá
ca. 900 A.D
AAEL VRC

Castillo
Toltec-Maya
ChichÃ©n ItzÃ¡
ca. 900 A.D
AAEL VRC

Move to UTF8

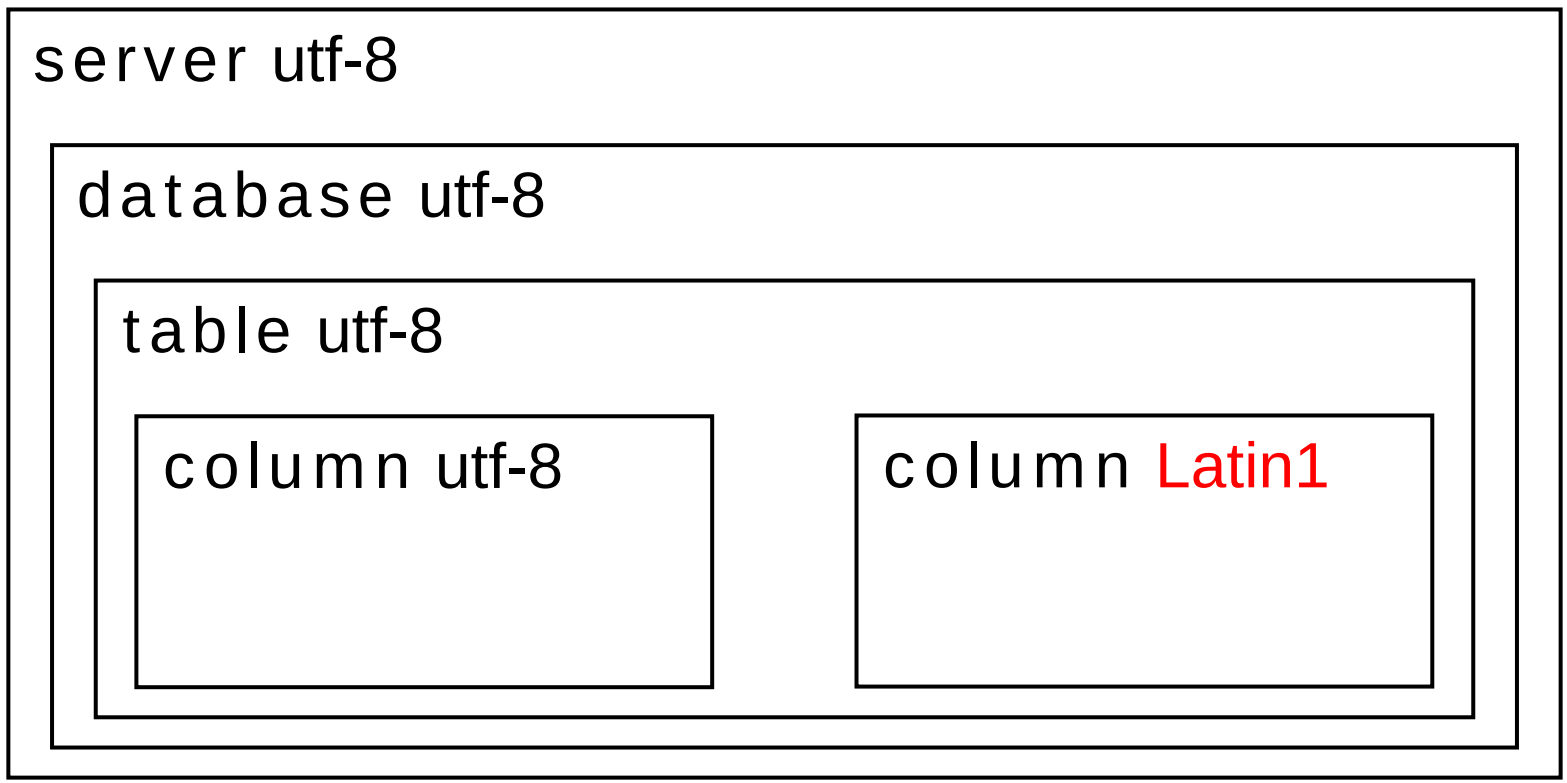
- Began with ASCII, Latin1, charents.
- Reloaded non-ASCII data as UTF-8.
- Loaded new/updated data as UTF-8.
- Left ASCII databases alone.

MySQL 4.1 Just In Time

- Robust character set support
- Minimal documentation

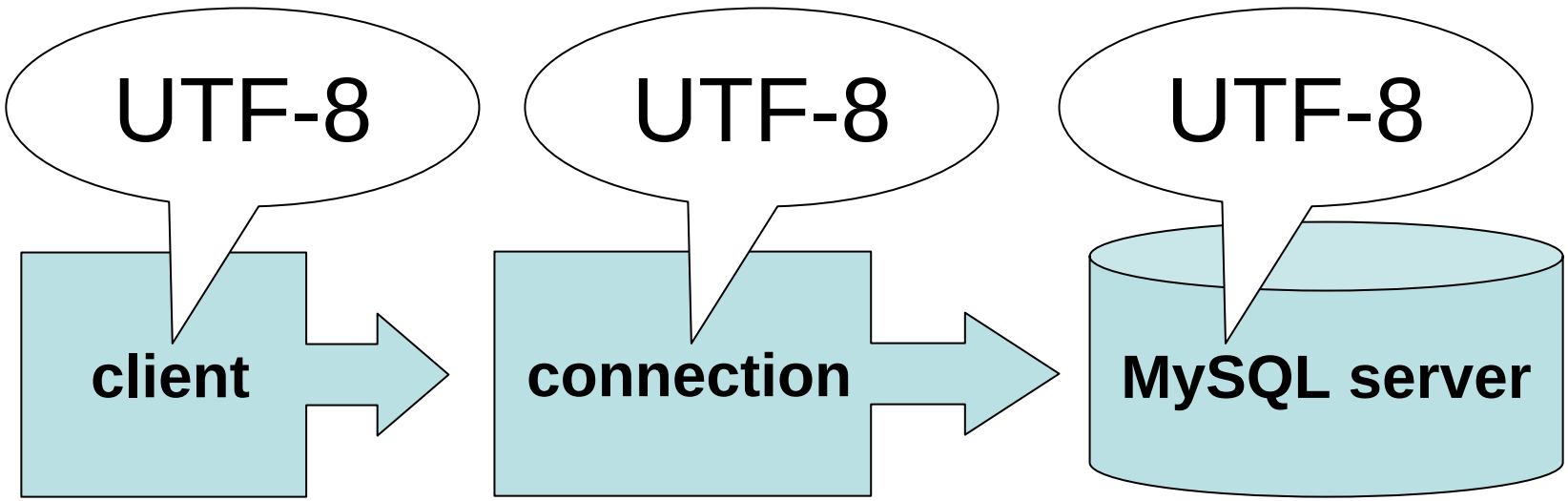
MySQL Server Character Set Support

Defined at every level, with inheritance.



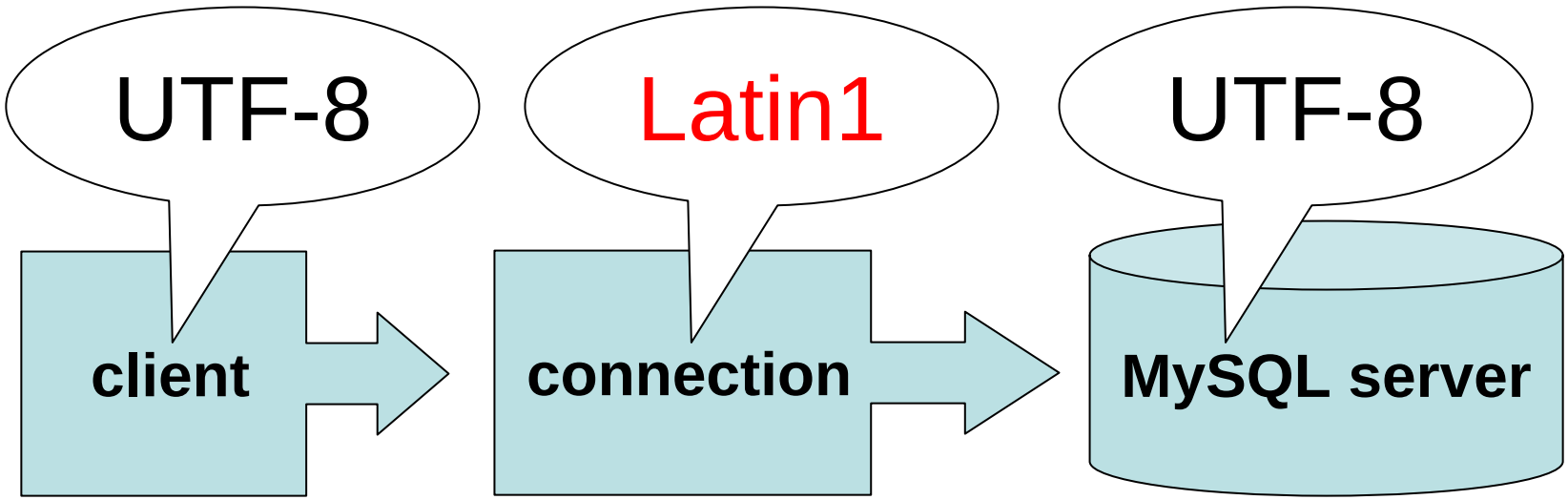
MySQL Connection Character Set Support (1)

- Reliable results depend on consistent communication between client and server.



MySQL Connection Character Set Support (2)

- Inconsistency introduces conversion that is sometimes lossy.



XPAT Background

- Proprietary search engine
- Source license from OpenText Corp.
- String index
- SGML region index
- Designed for single byte character encodings like iso-8859-1 (Latin1)

Unicode (in brief)

- Assigns a unique number to each character
- Defines several encodings for that number
- The Basic Multilingual Plane (BMP) covers 65,535 characters
- A BMP character occupies up to 3 bytes in the UTF-8 encoding
- So the size of a character in memory varies

XPAT software changes for Unicode

- Previously limited to 256 characters, i.e. one byte
- New internal storage 16 bit data type to store a character number up to 65,536
- New i/o routines to read bytes until a character was identified

XPAT configuration for Unicode

- Previously XPAT could support only 256 different characters
- Index points and mappings:

<IndexPt> &ISO_printable.</IndexPt>

<Map><From>\333</From><To>u</To></Map>

XPAT configuration for Unicode (cont.)

- Now: characters from different alphabets
- Unicode Block definitions define alphabets
- perl/lib/5.8.x/unicore/UnicodeData.txt
- perl/lib/5.8.x/unicore/Blocks.txt

<IndexPt> &Latin.</IndexPt>
<IndexPt> &Greek.</IndexPt>
<IndexPt> &Hebrew.</IndexPt>

<Map><From>U+00C0</From><To>U+0061</To></Map>
<Map><From>U+039F</From><To>U+03BF</To></Map>

Unicode in DLXS Middleware: Why?

- Unicode UTF-8 In / Unicode UTF-8 Out
- Common denominator for programming
- Common denominator for XML parsing
- Common denominator for characters in final HTML output
- <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

Unicode in DLXS Middleware (XPAT input)

- Most of our collection data has been converted to UTF-8 encoded Unicode
- So search results from XPAT are UTF-8
- Simply pass results directly to XML parser and write to STDOUT

Unicode in DLXS Middleware (XPAT Input cont.)

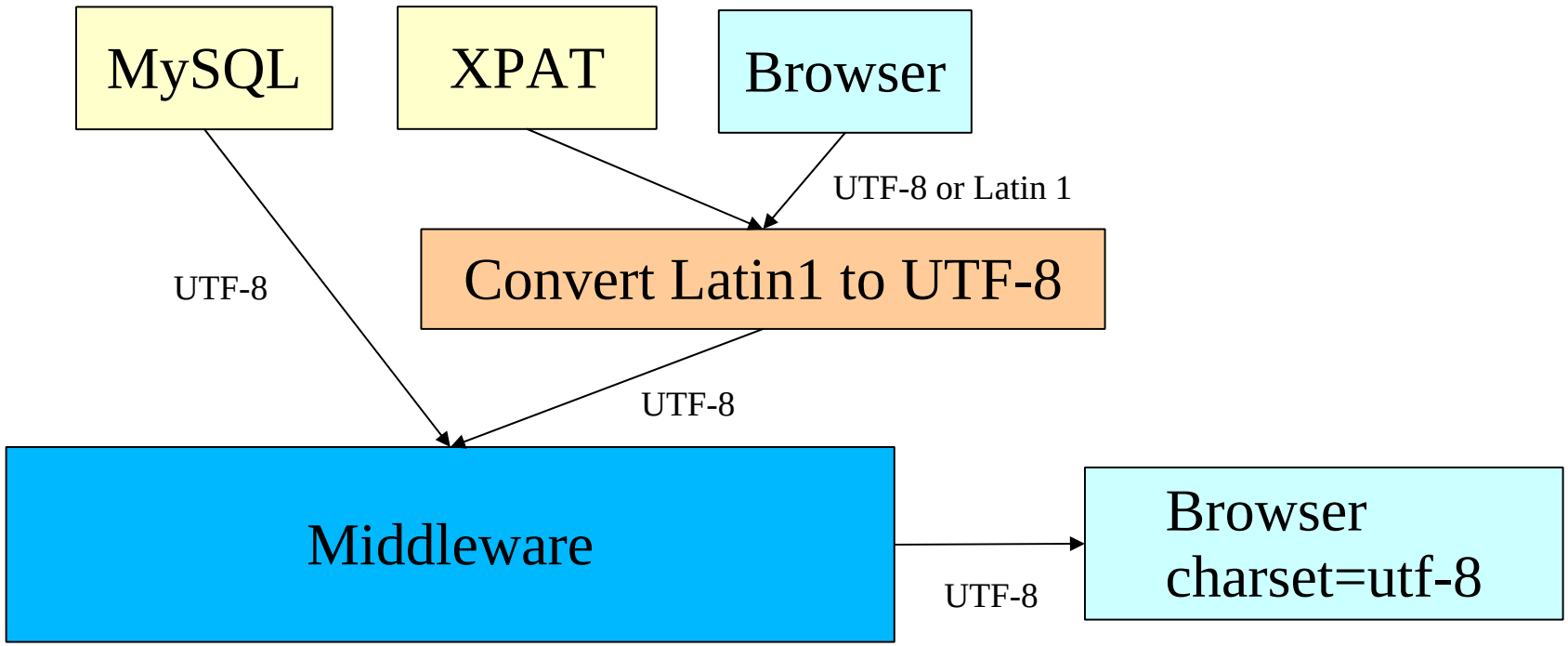
- Latin1 support as a migration path
- Conditionally convert XPAT Latin1 results to UTF-8 on the fly
- Optional inclusion of a Character Entity declaration in the XML before parsing -- é ℵ etc.

Unicode in DLXS Middleware (User input)

- All web forms have charset=UTF-8
- Still possible to receive non-UTF-8 input
- Test input string: if not UTF-8, assume Latin1
- Convert from Latin1 to UTF-8

Unicode in DLXS Middleware

- Goal: Inside the middleware all character data is UTF-8 encoded Unicode



Unicode in DLXS Middleware (Programming Perl)

- Perl 5.8.3 at least
- Perl must be told what encoding applies to its string data or it assumes Latin1
- UTF-8 flag tells Perl string is UTF-8
- UTF-8 flag propagates across concatenations, copying, etc.
- ... but there are problems beyond simple string operations...

Unicode in DLXS Middleware (Programming Perl cont.)

- Why UTF-8 Flag?
- So length, substring and matching in strings works on characters not bytes
- So Perl does not automagically convert your data to Latin1

Unicode in DLXS Middleware (Programming Perl cont.)

- When UTF-8 Flag?
- As early as possible when receiving input from XPAT and MySQL
- As late as possible when outputing user input stored in a CGI object because the flag does not propagate

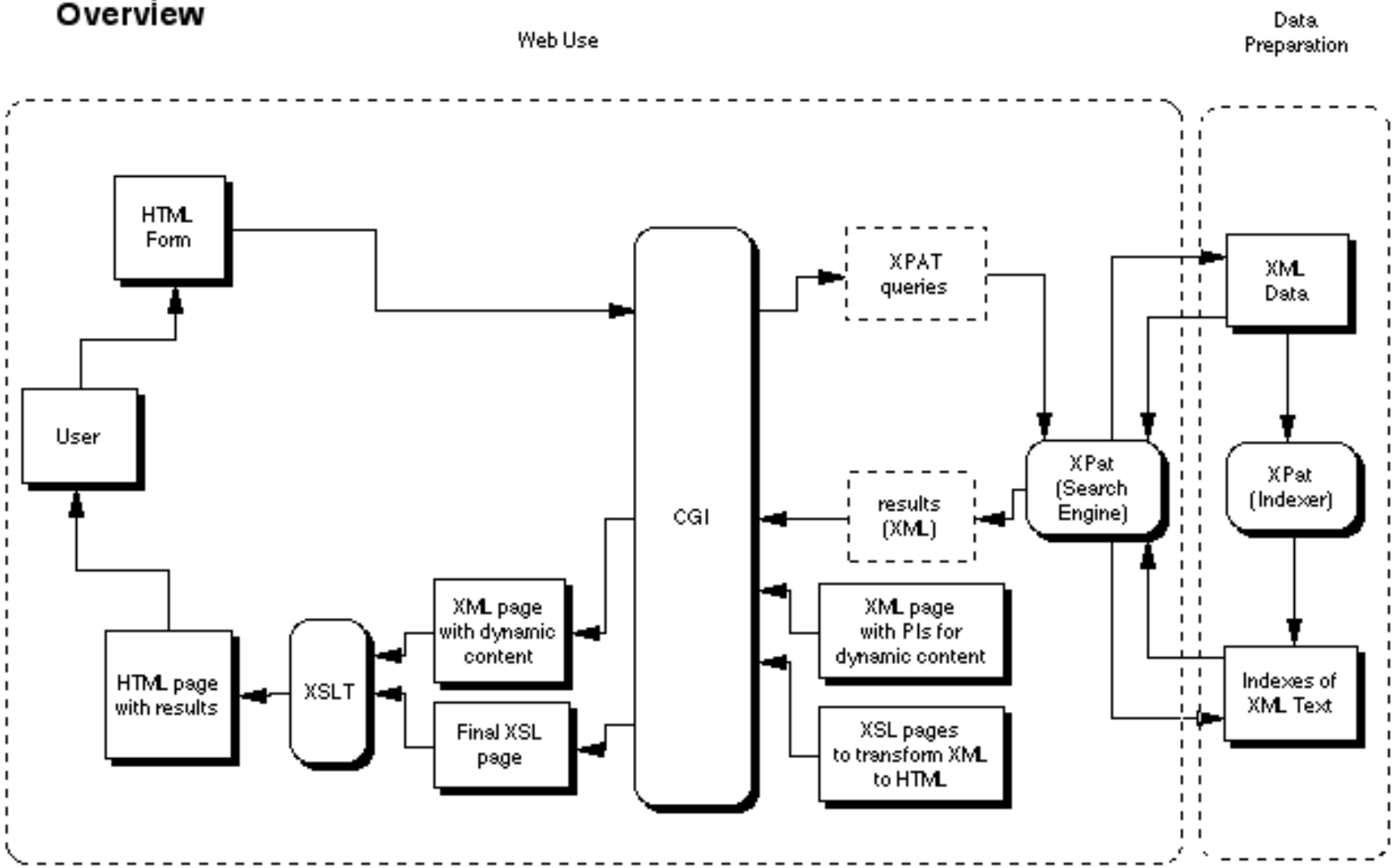
Unicode in DLXS Middleware (Programming Perl cont.)

- Programming lessons:
- Unicode UTF-8 in Perl still has bugs
- http://www.nntp.perl.org/group/perl.unicode/2787
- Some trial and error needed
- UTF-8 Flag does not always propagate

XML/XSLT in DLXS Middleware

- Bar napkin overview (see next slide)
- Getting well-formed XML out of XPat
- Learning XSL: programmers' perspectives
- XSLT engines, debuggers
- Division of labor between XSLT and CGI
- Virtual stylesheets
- Plan A, Plan B and why

Bar Napkin Overview



Getting Well-Formed XML from XPAT

- XPat results
  - Region sets
  - Point sets

XPat Region Result

```
<P><EPB/><PB REF="00000194.tif"
 SEQ="0194" RES="600dpi"
 FMT="TIFF5.0" FTR="UNSPEC"
 N="170"/>In going through the
 town... garments that were
 her own handiwork.</P>
```

XPat Point Result requires “Twigification”

```
F></ITEM><ITEM>proclamation of
 unity,<REF>xvii</REF></ITEM>
 <ITEM>Alexander, Prince, of
 Servia, <REF>179</REF></ITEM>
 <ITEM>Altgrafin, Political
 views
```

Learning XSL: Programmers' Perspective

- Syntax
- Processing
- Debugging
- Maintenance
  - Overall design
  - Modularity
  - Version tracking

XSLT Engines / Debugging

- Middleware
  - Perl XML::LibXML and XML::LibXSLT modules (wrappers for libxml and libxslt)
- Oxygen
  - XSLT debugger uses Saxon 6.5.4, 8B, 8SA or Xalan
  - Cannot be configured to use libxslt

Division of Labor

- Previously, Perl Middlware was responsible for converting the SGML/XML into HTML.
- Now
  - Perl Middleware
    - Controls application logic and link building
    - Emits well-formed XML
  - XSLT
    - Creates the HTML
    - User interface elements

Virtual Stylesheet

- Class / collection “look and feel”
- Run-time decision
- Problem XSLT 1.0 has no conditional importing of XSL stylesheets
- Workaround:
  - Perl Middleware builds top-level XSL file in memory

Project Management

- Timelines: need for flexibility
- Design decisions for system
- Interactions with other DLXS institutions
- Interactions with publishers of hosted content
- Testing
- Human resources

Surprises / Lessons Learned

- Lack of tools and documentation
  - Unicode: perl, text editors
  - XSLT debugger
- Workaround for fallback/XSL import
- Design and migration decisions
- Reworking XML DTD needed
- Race condition / XML file caching

Questions?

- Documentation: http://www.dlxs.org
- Contact: dlxs-help@umich.edu