





Development and deployment of Common Lisp applications on the JVM

Erik Huelsmann, ECLM2013

About me



- Owner of  HUCS B.V.:
Independent Business analyst (Financial) Reporting
- Co-owner of  **EFFICITO.COM**
Cloud-hosted open source ERP for SMB
- Open source developer of
(Subversion,) LedgerSMB
and **Armed Bear Common Lisp**

ABCL development progress



Milestones set in 2008

- Maxima test suite runs without crashing (achieved)
- Maxima test suite runs without errors (achieved)
- ANSI test suite runs without errors (<20 left out of 21.000)
- 1.0: CLHS compliance (last hurdle: support for long form of DEFINE-METHOD-COMBINATION)
- 2.0: AMOP compliance (estimated delivery: “far future”)

As a measure of practical usability



Development targets revisited

The world has changed due to

- Quicklisp
- CL-Test-Grid

→ Measuring *practical usability* redefined

ABCL development status



ABCL 1.2 release – ECLM 2013

- Closer to MOP compatible! (*Thanks Rudi!!*)
- Now runs lots of additional Quicklisp libraries
- Some long-standing bugs fixed (XPath)

Next steps

- Finally fix the pretty-printer / Gray streams interaction issues
- Work on optimizations (JAR size, execution)

ABCL use(r)s



Embedded:

- *Application engine*
Triggering events through Java UI,
Lisp business logic execution
- Scripting language
Java web application inside servlet container,
customization and trigger implementation

Commercial uses

Stand alone:

- (ask Hans Huebner)



Why use ABCL?

- Cross-platform UI
- Tap into VM research for JVM
- Tap into vast Java development resources
- To run inside the same image as Java application being interfaced with

But most of all:

- Customers like hearing: *It's Java based*

How?



StructuredXL

File Scenario Edit Inspect Help

Base Entity tree Element tree: PORT-P&L-2012 Period tree: PERIODS+FY

Scenario: DEFAULT Entity Element Period

Model - outcome

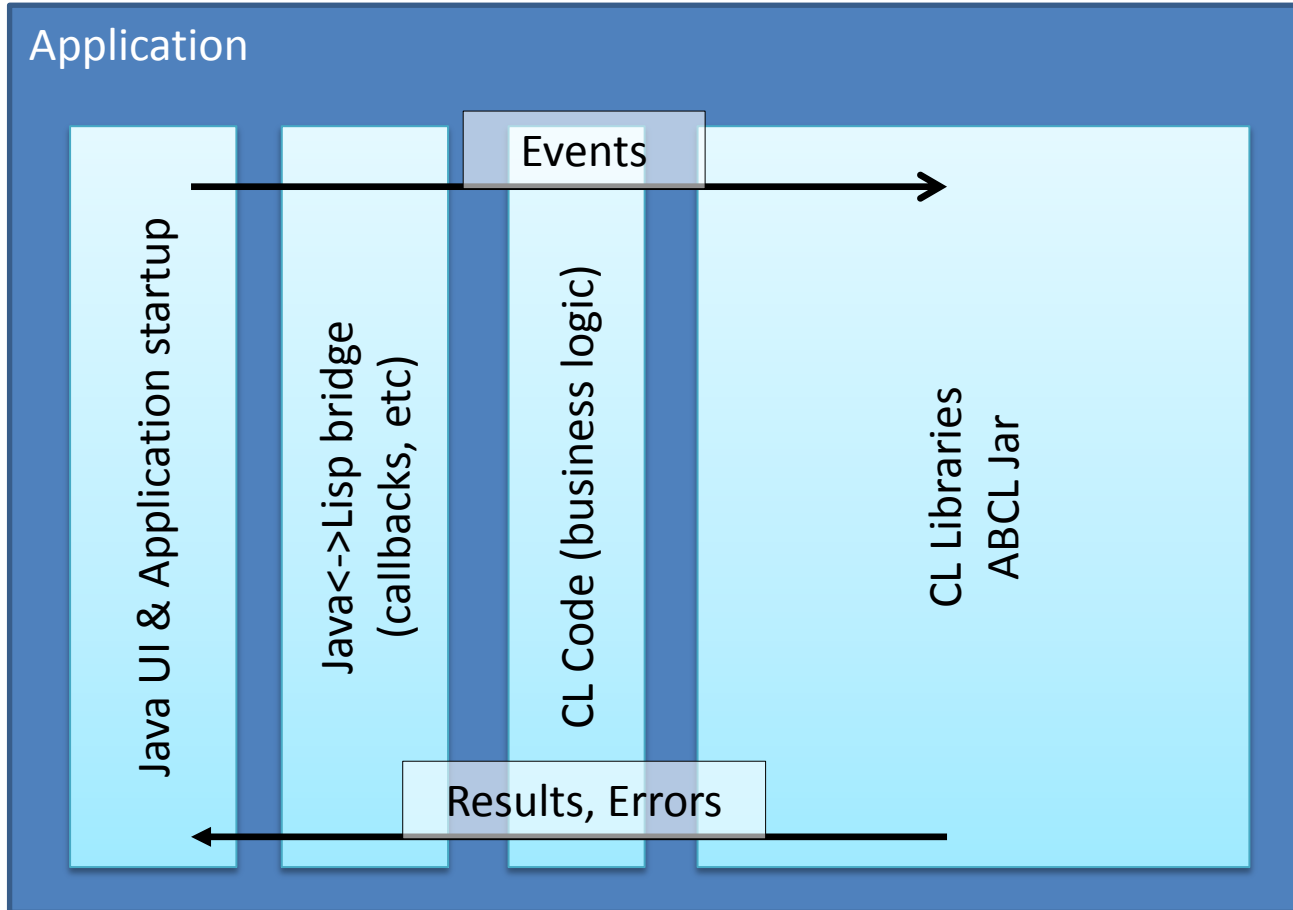
	2013FY	2013P01	2013P02	2013P03	2013P04	2013P05	2013P06	2013P07	2013P08	2013P09	2013P10	2013P11
PORTFOLIO												
PORTFOLIO-END-LAST-PERIOD												
PORTFOLIO-DEPOSITS-LAST-PERIOD												
PORTFOLIO-START												
NEW-PRODUCTION												
NEW-PRODUCTION-OLD												
NEW-PRODUCTION-GROWTH												
PRODUCTION												
EARLY-REDEMPTIONS												

Elements / years Elements / regions Entities / years

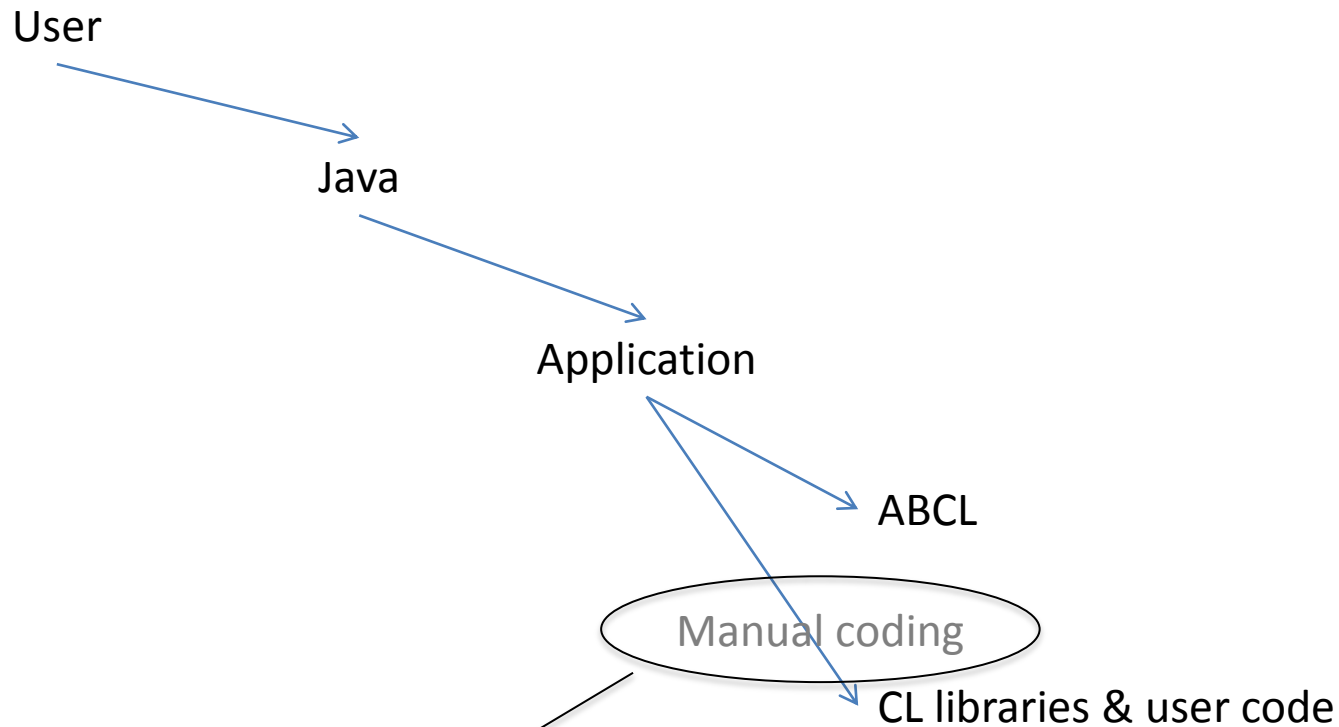
Changes	Change description	Change comment

Example: StructuredXL, Financial budgetting tool

Application structure



Application start-up



To be replaced with ASDFv3
precompiled concatenated
full system FASL

Build/Development tools



- NetBeans
UI design, event forwarding, bridge coding
- Emacs/SLIME
CL coding/testing (outside of app)
- Ant ('make' for Java)
Java compilation
- ABCL & ASDF (the ABCL application, not the library)
CL compilation (started from Ant)

Code samples (1/2)



Application start-up

```
100  /**
101   * Main method launching the application.
102   */
103  public static void main(String[] args) {
104      Interpreter.initializeLisp();
105      try {
106          File theJar = abclJar();
107          String thePath = java.net.URLDecoder.decode(theJar.getParent(), "UTF-8");
108
109          compileOrLoad(new String[] {
110              new File(thePath, "parse-number").getPath(),
111              "D:\\SXL\\budget-tool-wk26-branch\\lisp\\parse-number" },
112              false);
```

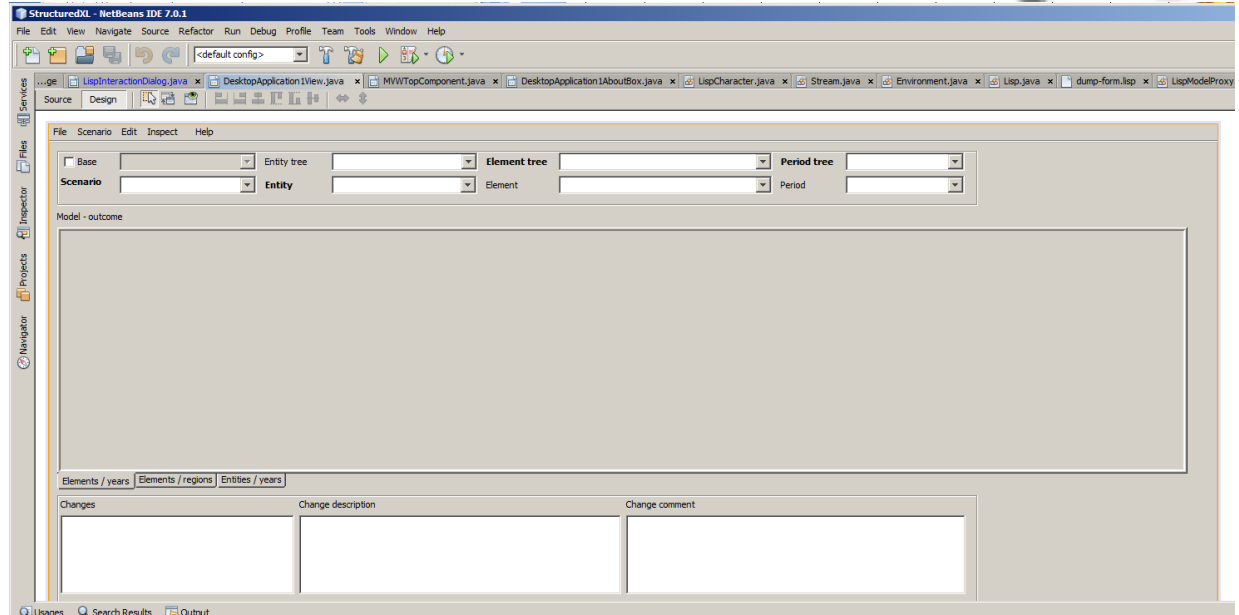
Java->CL bridge method

```
366  static LispObject createScenario(String name, String description)
367  {
368      return Lisp.funcall(makeScenario(),
369          new LispObject[] { keyword_name, new SimpleString(name),
370              keyword_description, (description == null) ? Lisp.NIL : new SimpleString(description) },
371          LispThread.currentThread());
372  }
```

Code samples (2/2)



NetBeans UI designer



The usual CL code

```
emacs@ACER
File Edit Options Buffers Tools Lisp Help
[Icons: Search Results Output]

(defun make-scenario (name &rest args
                    &key (model *model*)
                      anonymous
                      &allow-other-keys)
  (let ((final-args (cons :name (cons name args))))
    (remf final-args :anonymous)
    (let ((instance (apply #'%make-scenario final-args)))
      (unless anonymous
        (add-scenario model instance))
      instance)))
```

Deployment



Standard NetBeans packaging:

- Main application JAR
- Library JARs [abcl.jar, other java jars]
- CL fasl file(s) [* .abcl, in library directory]

Distribution

- Zipped or unpacked
 - Runs off any media including USB-stick
 - Total distribution size (incl. 8MB ABCL jar): 20MB

Servletcontainer deployment



Since time is probably up now,
see the examples in the repository...

That's it



Questions?