Micro-Cocoon

Running Cocoon in unusual places

Sylvain Wallez
sylvain@apache.org

Cocoon Get Together - 19 nov 2002

Agenda

- Introduction
- Needs for an embedded Cocoon
- Use cases
- Demonstration
- Which Cocoon is it ?
- How we did it

Introduction

sylvain@apache.org
Cocoon committer since may 2001

- Solved some nasty bugs in the XSP engine
  → That's how I became a committer
- Wrote the interpreted sitemap engine
- Introduced Writeable Sources
- Hacked a lot here and there ;-)
  → Most recent addition : ZipArchiveSerializer

Introduction

sylvain.wallez@anyware-tech.com
CTO at Anyware Technologies

- Located in Toulouse, France
- Co-founded in july 2000, now about 20 people
  → We use Cocoon since day one (v1.7.4 at that time)
- Web applications in the IT domain
  - Web Application Studio
  - XML/object persistence framework
- Web publishing of legacy or XML data
- XML, Cocoon & Java consulting and training

Needs for an embedded Cocoon

Connected appliances and terminals

- More and more networked appliances
  - Automation devices (industry)
  - Home gateways
  - Refrigerators, cars, etc.
- More and more ways to connect to appliances
  - Standard browser (HTML, SVG, PDF)
  - PDA browser (HTML, SVG-lite)
  - WAP phone (WML)
  - Voice phone (VoiceXML)

  → They're all XML based

Needs for an embedded Cocoon

Cocoon is the perfect solution

- Clean separation of presentation and content
- Easy multi-channel content production
- Integrates all XML-based formats
- Written in Java
  → not OS dependent
- Highly modular architecture
  → Can be tailored to specific needs

- But you already know all this
  - Otherwise you wouldn't be at this Get Together ;-)

Use cases

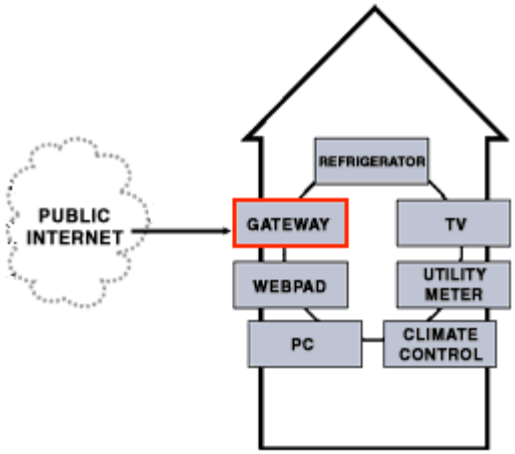
Industrial automation

- In-factory monitoring and control
  - → Regular HTMl pages
  - → Dynamic graphics with SVG
- People on duty at home (24/24 service)
  - → Web browser
  - → Wap phone (WML)
  - → Voice phone (VoiceXML)

Use cases

Home gateways

- Central hub of house equipments
- Inside the house : PC-based
  - HTML & SVG
  - SOAP interface
    - → PC with a .Net GUI
- Outside the house
  - Monitoring
  - Remote control
  - → Web browser (at work)
  - → Wap/voice phone



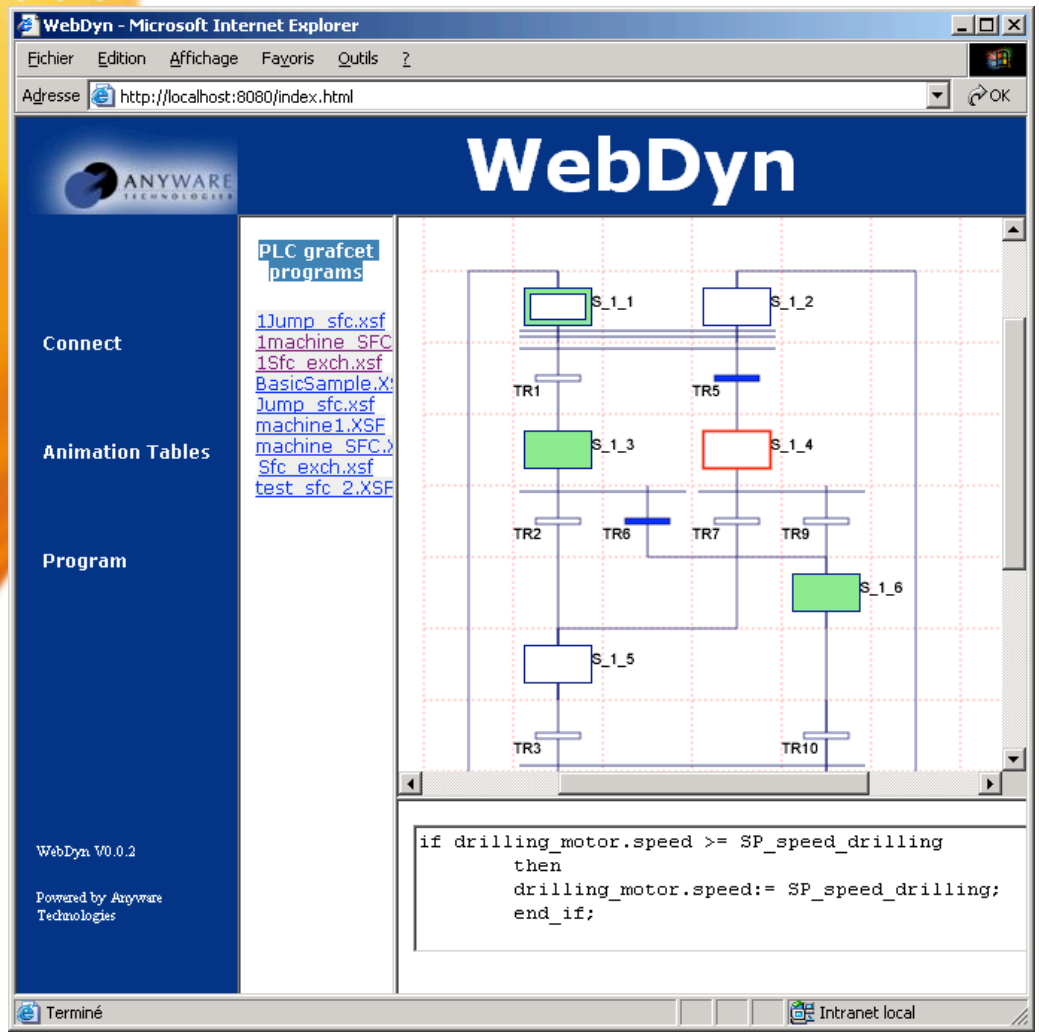
Use cases

Refrigerators

- Front-door : LCD panel with HTML browser
  - → Manage expiry dates
  - → Order from online shops
  - → Consult/print the cookbook
- In the supermarket : wap phone
  - → Know what's inside to know what to buy

Demonstration

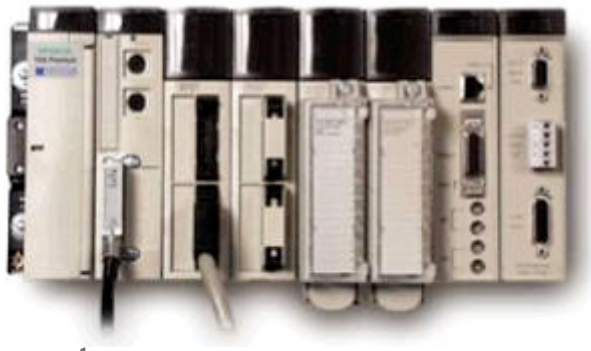
Industrial process monitoring



- Program textual and graphical display (SVG)
- Program animation
- HTML view of monitored data
- Excel export of monitored data
- Wap display of monitored data

Demonstration

The target device



- Hardware environment
  - 50 MHz PowerPC
  - 32 Mbytes memory
- Software environment
  - VxWorks real-time operating system
  - PersonalJava virtual machine
- The application
  - 1.1 Mbytes jar (includes everything)
  - A few files (data, XSLT and static content)
- Average response time < 300 ms

  → And still much room for improvement !!

Which Cocoon is it ?

It's a standard Cocoon !

- Standard 2.0.3 distribution
- Uses the compiled sitemap engine
  - Too bad, I'm the one who killed it :-(
  - Interpreted engine adds 250 kb of bytecode

  → We are likely to write a new, more modular, compiled engine
- Sitemap and XSPs are precompiled
  - Use of Cocoon's command line features

Which Cocoon is it ?

A few custom components

- SAX only XML Parser
  → Because most applications don't use DOM
- Source-less ProgramGenerator (XSP engine)
  → Because XSPs and sitemap are precompiled
- Memory-only store
  - Not backed by a persistent store

  → Because there's no hard-disk !

Which Cocoon is it ?

Ultra-light servlet engine

- Less than 100 kb of bytecode
- Some restricted features…
  - Single webapp context
  - No automatic class reloading
- …but everything that is needed by a Cocoon app !

Which Cocoon is it ?

XSLT processing

- The most memory and CPU consuming task
- Currently : SAXON
  - Includes the Ælfred parser (small and SAX-only)
  - Quite fast
  - About 900 kb of bytecode
- In the works : XSLTC
  - Precompile all XSLTs to bytecode
  - Small and fast runtime
- Alternative : STX (Streaming Transformations for XML)
  - XSLT-like language for SAX-based transformations
  - See http://stx.sf.net for more info

How we did it

Compatibility problem

- Small devices run limited JVMs
  - Currently : PersonalJava
  - Soon : J2ME CDC
- What is PersonalJava ?
  - A stripped-down JDK 1.1.8 with some JDK 1.2 security classes
- Cocoon cannot run on PersonalJava
  - Missing classes
    - → e.g. collections in `java.util`
  - Missing methods on existing classes
    - → e.g. `File.toURL()`

How we did it

How to adapt Cocoon to PersonalJava ?

- Solution 1 : do it manually
  - Add some re-implementations of missing classes
    - → GNU classpath project
  - Modify the source code where it isn't compatible

→ Tedious, error prone and to be done again for each new Cocoon version

- Solution 2 : do it automatically
  - Analyze the call graph and "import" missing classes
  - Transform the bytecode to remove incompatibilities

How we did it

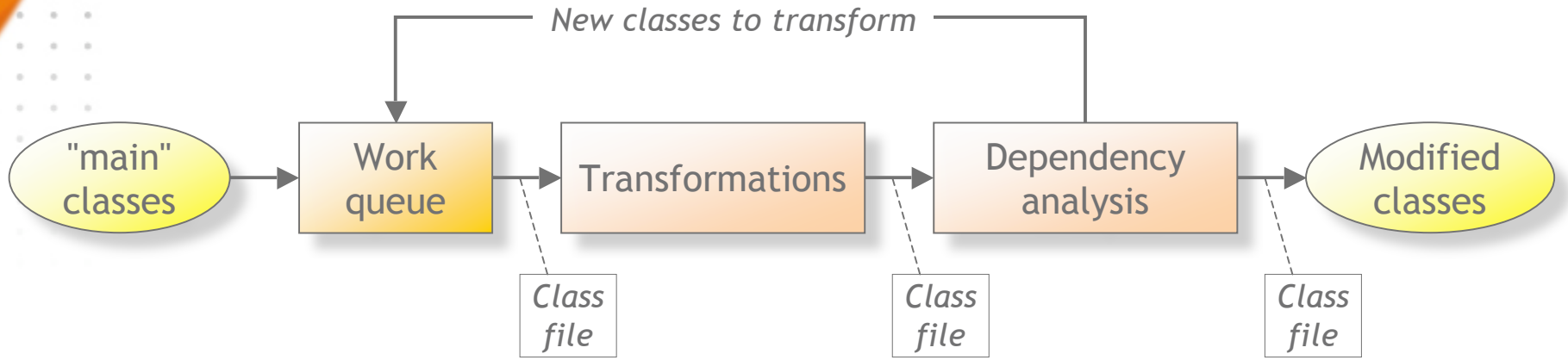
Introducing J-Fit, to make Java really run anywhere

- Bytecode analysis and transformation
  → Based on Jakarta BCEL
- Inputs
  - Runtime classes from your JDK (e.g. JDK 1.3.1)
  - Your application classes and libraries
  - Runtime classes from the target JDK (e.g. PJava)
  - Names of the main and dynamically loaded classes
    → Automatic analysis of Cocoon's xconf and xmap files
- Outputs
  - A compatible ready-to-deploy jar file !
  - Call graph analysis report

How we did it

J-Fit : pipelined class transformations

- Set of transformation components
  - → Avalon-based
  - → Define your own pipeline !
- Closed loop feedback
  - Call graph analysis to find new classes to be transformed



How we did it

J-Fit : transformer samples

- Translate non-existing methods

```
URL fileURL = file.toURL();
```

```
URL fileURL = FileCompat.toURL(file);
```

- Remove debug statements

  → debug = a lot of code and Strings

```
if (getLogger().isDebugEnabled())
  getLogger().debug("I was here !");
```

Conclusion

Cocoon can run (almost) anywhere

- A new range of opportunities for networked devices
  - Easy building of multi-channel applications
  - Can provide a lot of new features

- Some limitations, however
  - → Memory : choose used components wisely
  - → Compatibility : tools like J-Fit greatly help

The end

Questions ?
Answers !