http://bitser.net/isolate-interest

BOF-2436: The Java^{tm} Isolation API

Past, Present and Future

Bernd Mathiske
Sun Microsystems
bernd.mathiske@sun.com

Pete Soper
Sun Microsystems
pete.soper@sun.com

June 29, 2004

Isolate *noun*. pronounciation: ***isolet***. 1. A thing that has been isolated, as by geographic, ecologic or social barriers
- *American Heritage Dictionary*

- Regrets from Doug Lea
- Motivation
- API overview and code examples
- Status
- Over to Bernd

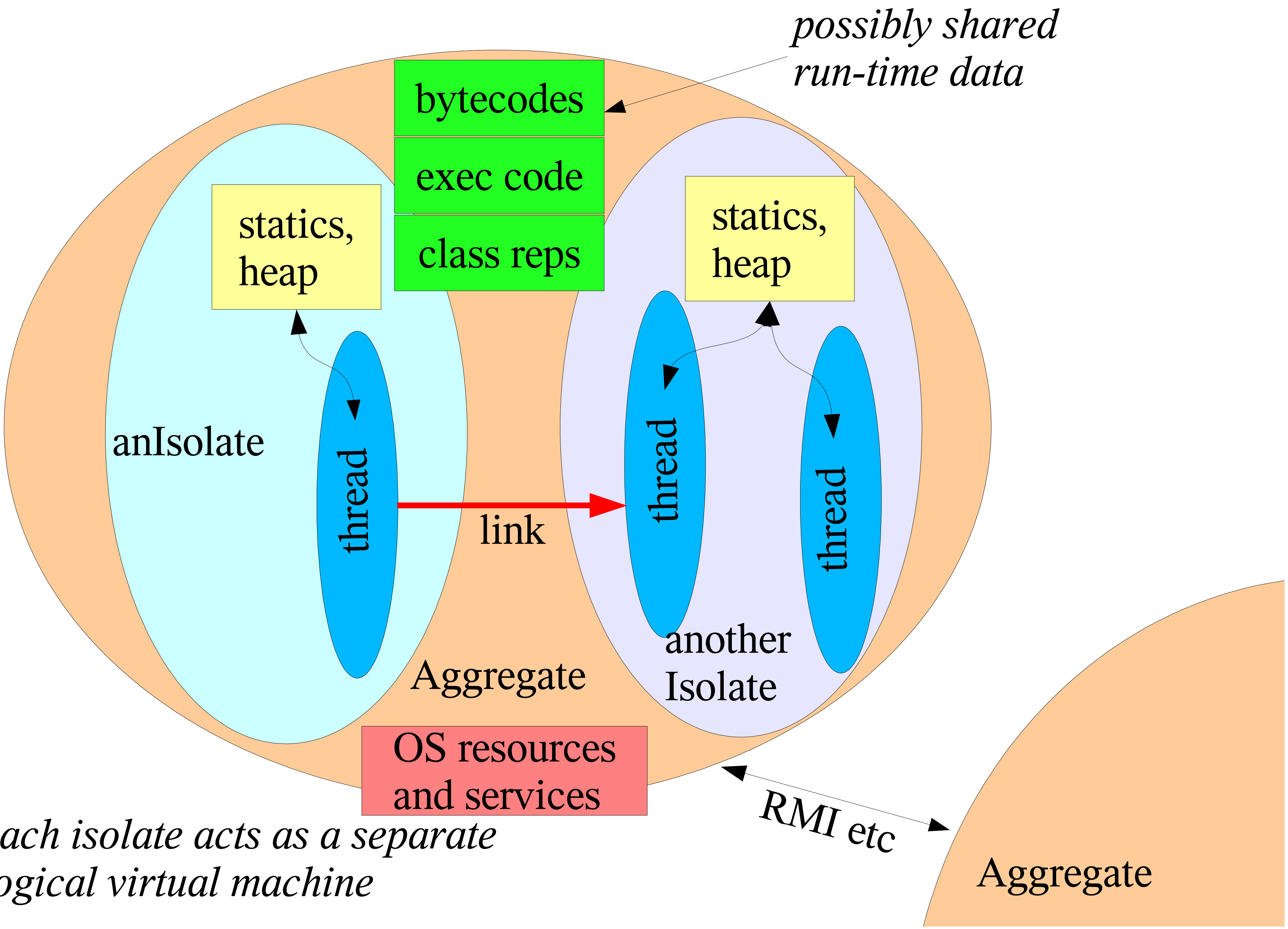
Motivation

- Eliminate unintended sharing of state
  - Alternative to classloader scoping to achieve*complete* isolation
- Define unit of manageability
  - Life cycle control (now)
    - Configure, monitor and kill activities without disrupting others
  - Resource management (later)
- Combine safety with scalability
- Security
  - Simpify construction of obviously secure systems
  - Stay within Java (vs leaving/reentering via Runtime.exec)

API Design Goals

- Minimality
  - The smallest API that fills need
- Mechanism, not policy
  - Enable layered frameworks
- Simple, clean semantics
  - For termination, communication, etc
- Compatibility
  - No changes required in pre-JSR-121 code
- Generality
  - Allow multiple mapping strategies to platforms

Aggregates vs Isolates vs Threads



each isolate acts as a separate logical virtual machine

Implementation Styles

- One Isolate per OS process
  - Internal sharing via OS-level shared memory, comms via IPC
    - class representations, bytecodes, compiled code, immutable statics, other internal data structures

“Simple RI”

- All Isolates in one OS address space / process
  - Isolates still get own versions of all statics/globals
    - including AWT thread, shutdown hooks, ...

MVM, Janos VM

- Isolates scheduled onto JVMs

SAP Research

- LAN Cluster JVMs
  - Isolates on different machines, one admin domain.

API Structure (base package)

- Package javax.isolate
  - Isolate
  - IsolateParameters
  - Link
  - DataMessage
  - StatusMessage
  - CompositeMessage
- New Interface
  - Message (just a tag)
- New Exceptions
  - IsolateStartupException
- Changes to existing APIs
  - Documentation clarifications

API Structure (additional pkgs)

- javax.isolate.tbd (CDC+)
  - IsolatePermission
  - ObjectMessage
- javax.isolate.io (J2SE)
  - IOMessage interface
  - file/network I/O classes
- javax.isolate.nio (J2SE)
  - ByteBuffer
  - ChannelMessage
- javax.isolate.util (J2SE)
  - Visitor pattern & support

Main Classes

- **`public final class Isolate implements Message`**
  - Create with name of class with a "**`main`**", arguments (simple) or with IsolateParameters (two flavors of additional parms)
  - Methods to start and terminate and query isolate, get its parms and starting links
- **`public class Link`**
  - A pipe-like data channel to another isolate
    - byte arrays, ByteBuffers, Strings and serializable types
    - SocketChannels, FileChannels and other IO types
    - Isolates, Links

Starting Isolates

- Isolate creation establishes existence
  - Isolates may (but need not) perform resource allocation and internal initialization upon creation
- Static initializers, then main run at **`start`**
  - Isolates may continue initialization before running
  - All classes are loaded in new Isolate's context
- Failures detected before running user code result in exceptions at creation or start time
  - Cannot be sure whether the same exceptions will be thrown at the same points in all Implementations
- Other failures merely terminate the Isolate

Running Independent Programs

```
void runProgram(String classname,
                String[] args) {
  try {
    new Isolate(classname, args).start();
  }
  catch (SecurityException se) { ... }
  catch (IsolateStartException ise) { ... }
  catch (Exception other) { ... }
}
```

Initializing and Monitoring

```
Class Runner {
  Link data;
  Isolate child;
  CompositeMessage getMessage() { return data.receive(); }
  StatusMessage runStarlet(String mCls, String[] mArgs,
                              String[] sec /*,...*/) {
    IsolateParameters context = new
                                 IsolateParameters(mCls, mArgs);
    context.setContext(
       “jsr121.exp.java.properties.java.security.manager”,
            sec);
    child = new Isolate(context);
    data = Link.newLink(child, Isolate.currentIsolate());
    StatusLink s = child.newStatusLink();
    child.start(new Link[] { data } );
    return s.receive();
  }
}
```

Status

- JSR 121 page at the JCP
  - http://jcp.org/jsr/detail/121.jsp
- Isolate-interest mailing list
  - http://bitser.net/isolate-interest/
- Bibliography of related work
  - http://www.bitser.net/isolate-interest/bib.html
- First public review implementations
  - http://www.cs.utah.edu/flux/janos/
    - Partial, no NIO
    - Derived from Kaffe, pre-Java2, strictly speaking not Java™
    - “many isolates to one JVM style”
  - Feature complete on two platforms, not included in J2SE 1.5
- APIs refactored and moved to javax.

Next Steps

- Upgrade JSR-121 to JCP rev 2.6
- Involve Community
  - Expand EG
  - Tap into java.net if possible
  - Exercise, explore use cases and***validate*** API design with the most transparent process possible
  - Start with simplest package layers (resources, demand drive priorities)
- Second Public Review
- Finish spec(s), RI(s) and TCK(s)
  - Maybe weak binding with Sun release cycles
  - Deliver to J2SE and J2ME process at the same time