RAPID NETWORK APPLICATION DEVELOPMENT
WITH APACHE MINA

Trustin Lee
Principal Software Engineer
Red Hat, Inc.

TS-4814
Learn how to build:

scalable, stable, maintainable and manageable network applications utilizing any protocol

with Apache MINA
Agenda

Before the adventure...

- Presenter
- Introduction
- Core Components
- Management
- Future
- Summary
Presenter
Who is Trustin Lee?

- Founder of Netty framework
- Cofounder and VP of Apache MINA
- JBoss Remoting project lead
- Wrote Java™ New I/O API (NIO)-based massive network applications
  - Distributed SMS gateway – 10M msgs / day
  - OSGi-based asynchronous RPC server with Hessian protocol
- Didn't write a book yet! ;)

Didn't write a book yet! ;}
Agenda
What, Why and How?

- Presenter
- Introduction
- Core Components
- Management
- Future
- Summary
Introduction

What is Apache MINA?

- A Java open-source network application framework

- Abstract API
  - Event-driven
  - Asynchronous
  - Unit-testable

- Implementations
  - Sockets & datagrams – Java NIO & APR via Tomcat Native
  - Serial ports – RXTX.org
  - In-VM pipes
  - <Your favorite transports here: SCTP, multicast, Infiniband...>
Introduction

Why should I use it?

> Maintainable and reusable
  - Networking engine – MINA I/O service
  - Protocol codec – MINA codec framework
  - Your business logic

> Extensible
  - Runtime modification of application behavior using 'filters'

> Manageable
  - Introspection of connections and services via JMX™ API

> Unit-testable
  - Abstract API
  - Out-of-the-box mock objects
Introduction

What does it look like?

- Performs actual I/O
- Filters events & requests
- A connection
- <Your protocol logic>
Agenda
Let’s learn by looking at examples!

- Presenter
- Introduction
- Core Components
- Management
- Future
- Summary
IoSession & IoBuffer
Writing a message was never easier than this.

// Build a string to send.
CharsetEncoder = ...;

// Build a string to send.
CharsetEncoder = ...;

IoSession session = ...;

IoBuffer buffer = IoBuffer.allocate(16);

buffer.setAutoExpand(true)
    .putString("It is ", encoder)
    .putString(new Date().toString(), encoder)
    .putString(" now.\r\n", encoder).flip();

// Asynchronous write request.
session.write(buffer);
IoSession

Connection, Socket, Channel...

- Abstracts a underlying transport’s connection away
- Provides asynchronous operations to I/O service
  - Write, close...
  - All asynchronous
  - Returns IoFuture (WriteFuture, CloseFuture...)
    - A set of IoFutureListener can be added for notification
- Provides I/O statistics
  - Read bytes, written bytes, last I/O time...
IoBuffer

Why don’t you just use NIO ByteBuffer?

- Rich binary & text manipulation methods
  - Unsigned value, enum, string, Java Object...
- On-demand automatic expansion and shrinkage
- More control over allocation mechanism
- More extensible than ByteBuffer
  - provides all methods in ByteBuffer
  - provides easy wrap · unwrap methods
IoHandler
Let’s write back what’s received.

```java
public class EchoHandler implements IoHandler {
    public void messageReceived(IoSession s, Object msg) {
        IoBuffer buffer = (IoBuffer) msg;
        s.write(buffer.duplicate());
    }

    public void exceptionCaught(IoSession s, Throwable e) {
        s.close();
    }

    public void sessionOpened(IoSession s) {}
    public void messageSent(IoSession s, Object msg) {}
    public void sessionIdle(IoSession s, IdleStatus stat) {}
    public void sessionClosed(IoSession s) {}
}
```
IoService
IoAcceptor is for the server side.

```java
public class Main {
    public static void main(String[] args) {
        IoAcceptor acceptor = new NioSocketAcceptor();
        acceptor.setHandler(new EchoHandler());
        acceptor.bind(new InetSocketAddress(8080));
        ... 
        acceptor.unbind(new InetSocketAddress(8080));
    }
}
```
IoService

IoConnector is for the client side.

```java
public class Main {
    public static void main(String[] args) ... {
        IoConnector connector = new NioSocketConnector();
        connector.setHandler(new MyHandler());
        ConnectFuture future = connector.connect(
            new InetSocketAddress("example.com", 8080));

        IoSession session = future.await().getSession();

        session.write(...).await(); // WriteFuture
        session.close().await();    // CloseFuture
    }
}
```
IoService
Switching to a different transport was never easier than this.

```java
IoAcceptor acceptor = new NioSocketAcceptor();
IoAcceptor acceptor = new AprSocketAcceptor();
...

IoConnector connector = new NioSocketConnector();
IoConnector connector = new SerialConnector();
...
connector.connect(new InetSocketAddress(...));
connector.connect(new SerialAddress(...));
...
```
IoFilterChain & IoFilter
Imagine hot-deployable Servlet filters.

// Enable logging.
acceptor.getFilterChain().addLast("logger", new LoggingFilter());

// Enable SSL.
acceptor.getFilterChain().addLast("ssl", new SslFilter());

// Enable compression for an individual session.
session.getFilterChain().addBefore("ssl", "compressor",
    new CompressionFilter());

// Zap all of them.
session.getFilterChain().clear();
IoFilter
One-stop solution for cross-cutting concerns:

- Logging
- Overload prevention
- Failure injection
- On-demand profiler
- Remote peer blacklisting
- Keep-alive · timeout
- More to come – whatever you want to intercept!
Protocol Codecs

Why do we need a protocol codec?

➢ It is a bad idea to implement a protocol only with IoBuffers.
  • Packet fragmentation and assembly
  • Separation of concerns

➢ Codecs are often reusable – MINA provides:
  • Text line codec
  • Object stream codec
  • HTTP codec

➢ MINA also provides reusable components to build a codec.
  • Solutions for packet fragmentation and assembly issue
  • Finite state machine framework dedicated to codec construction
  • Support for multi-layered protocol (e.g. Kerberos)
Protocol Codecs
What does it look like with a protocol codec?

- POJO → IoBuffer
- IoBuffer → POJO
Protocol Codecs

Echo server redux – TextLineProtocolCodecFactory kicks in!

```java
public class EchoHandler extends IoHandlerAdapter {
    public void messageReceived(IoSession s, Object m) {
        s.write((String) m);
    }
    ...
}
...
acceptor.getFilterChain().addLast("codec", new ProtocolCodecFilter(
        new TextLineCodecFactory()));
...
Protocol Codecs

Custom AJAX-ready HTTP server in 10 minutes!?

```java
public class HttpHandler extends IoHandlerAdapter {
    public void messageReceived(IoSession s, Object msg) {
        HttpRequest req = (HttpRequest) msg;
        MutableHttpResponse res = new DefaultHttpResponse();
        IoBuffer content = ...;
        res.setContent(content);
        res.normalize(req);
        s.write(res);
    }
}
...
acceptor.getFilterChain().addLast("codec",
    new ProtocolCodecFilter(new HttpProtocolCodecFactoryFactory()));
...
Thread Models
It’s as easy as inserting an IoFilter.

// Single thread model by default.
...

// One thread pool - suitable for typical servers.

//// Place CPU-bound tasks first,
acceptor.getFilterChain().addLast("compression", ...);
acceptor.getFilterChain().addLast("codec", ...);

//// And then thread pool.
acceptor.getFilterChain().addLast(
    "executor", new ExecutorFilter(
        new OrderedThreadPoolExecutor(16)));

//// Use UnorderedThreadPoolExecutor or your favorite
//// Executor instance if you don't want event ordering.
Agenda

JMX integration – brain-dead easy!

- Presenter
- Introduction
- Core Components
- Management
- Future
- Summary
Management

IoService, IoSession and IoFilter are exposed as JMX MBean.

```java
MBeanServer mbs = ...;

mbs.registerMBean(new IoServiceMBean(acceptor),
        new ObjectName(...));

mbs.registerMBean(new IoSessionMBean(session),
        new ObjectName(...));

mbs.registerMBean(new IoFilterMBean(executorFilter),
        new ObjectName(...));
```
Management

What you can do in runtime with MINA MBeans:

- Monitor various performance counters
- Adjust all socket parameters
- Start · stop an IoService
- Modify an IoSession based on OGNL expression
  - Find all session originating from '192.168.0.x' and close them all!
- Insertion and removal of an IoFilter
  - Enable or disable whatever on demand!
    - Logging
    - Profiling
    - Changing thread model
### Attribute values

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>activationTime</td>
<td>Fri Apr 04 21:03:45 KST 2008</td>
</tr>
<tr>
<td>active</td>
<td>true</td>
</tr>
<tr>
<td>backlog</td>
<td>50</td>
</tr>
<tr>
<td>bothIdle</td>
<td>false</td>
</tr>
<tr>
<td>bothIdleCount</td>
<td>0</td>
</tr>
<tr>
<td>bothIdleTime</td>
<td>0</td>
</tr>
<tr>
<td>bothIdleTimeInMillis</td>
<td>0</td>
</tr>
<tr>
<td>closeOnDeactivation</td>
<td>true</td>
</tr>
<tr>
<td>cumulativeManagedSessionCount</td>
<td>1</td>
</tr>
<tr>
<td>defaultLocalAddresses</td>
<td>*:0</td>
</tr>
<tr>
<td>disposed</td>
<td>false</td>
</tr>
<tr>
<td>disposing</td>
<td>false</td>
</tr>
<tr>
<td>filterChainBuilder</td>
<td>{codec=org.apache.mina.filt...</td>
</tr>
<tr>
<td>handler</td>
<td>JmxTest$1</td>
</tr>
<tr>
<td>largestManagedSessionCount</td>
<td>1</td>
</tr>
<tr>
<td>largestReadBytesThroughput</td>
<td>2.6657780739753414</td>
</tr>
<tr>
<td>largestReadMessagesThroughput</td>
<td>0.3333333333333333</td>
</tr>
<tr>
<td>largestWrittenBytesThroughput</td>
<td>2.332555814728424</td>
</tr>
<tr>
<td>largestWrittenMessagesThroughput</td>
<td>0.3333333333333333</td>
</tr>
</tbody>
</table>
### Attribute values

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bothIdle</td>
<td>false</td>
</tr>
<tr>
<td>bothIdleCount</td>
<td>0</td>
</tr>
<tr>
<td>closing</td>
<td>false</td>
</tr>
<tr>
<td>config.bothIdleTime</td>
<td>0</td>
</tr>
<tr>
<td>config.bothIdleTimeInMillis</td>
<td>0</td>
</tr>
<tr>
<td>config.keepAlive</td>
<td>false</td>
</tr>
<tr>
<td>config.maxReadBufferSize</td>
<td>65536</td>
</tr>
<tr>
<td>config.minReadBufferSize</td>
<td>64</td>
</tr>
<tr>
<td>config.oobInline</td>
<td>false</td>
</tr>
<tr>
<td>config.readBufferSize</td>
<td>128</td>
</tr>
<tr>
<td>config.readerIdleTime</td>
<td>0</td>
</tr>
<tr>
<td>config.readerIdleTimeInMillis</td>
<td>0</td>
</tr>
<tr>
<td>config.receiveBufferSize</td>
<td>43744</td>
</tr>
<tr>
<td>config.reuseAddress</td>
<td>true</td>
</tr>
<tr>
<td>config.sendBufferSize</td>
<td>25146</td>
</tr>
<tr>
<td>config.soLinger</td>
<td>-1</td>
</tr>
<tr>
<td>config.tcpNoDelay</td>
<td>false</td>
</tr>
<tr>
<td>config.throughputCalculationInteger</td>
<td>3</td>
</tr>
<tr>
<td>config.throughputCalculationInteger</td>
<td>3000</td>
</tr>
</tbody>
</table>
Agenda

A lot more to come!

- Presenter
- Introduction
- Core Components
- Management
- Future
- Summary
Future

Major tasks ahead:

> Zero copy I/O
  - Looking for better alternative to IoBuffer

> IoConnector improvements
  - Proxy support – patch pending
  - Automatic reconnection

> Better documentation

> Protocol codec generator
  - Rapid legacy & new protocol implementation

> Tools based on a protocol codec implementation
  - Protocol analyzing proxy
  - Intelligent L7 switch & firewall
Agenda

So, what’s the verdict?

- Presenter
- Introduction
- Core Components
- Management
- Future
- Summary
Summary

Apache MINA is designed exactly for:

- Any kind of network applications
  - Stable
  - Scalable
  - Extensible
  - Manageable
  - Unit-testable

- Simple, complex, text, binary, legacy and evolving protocols

- You got to try it now! ;)
For More Information
Vibrant community – that's what we are.

➤ WWW  –  MINA.apache.org

➤ E-mail  –  users@mina.apache.org
  trustin@apache.org (me)

➤ Please talk to me right after this session.