



JavaOne™

java.sun.com/javaone

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

GOAL

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! ;)



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

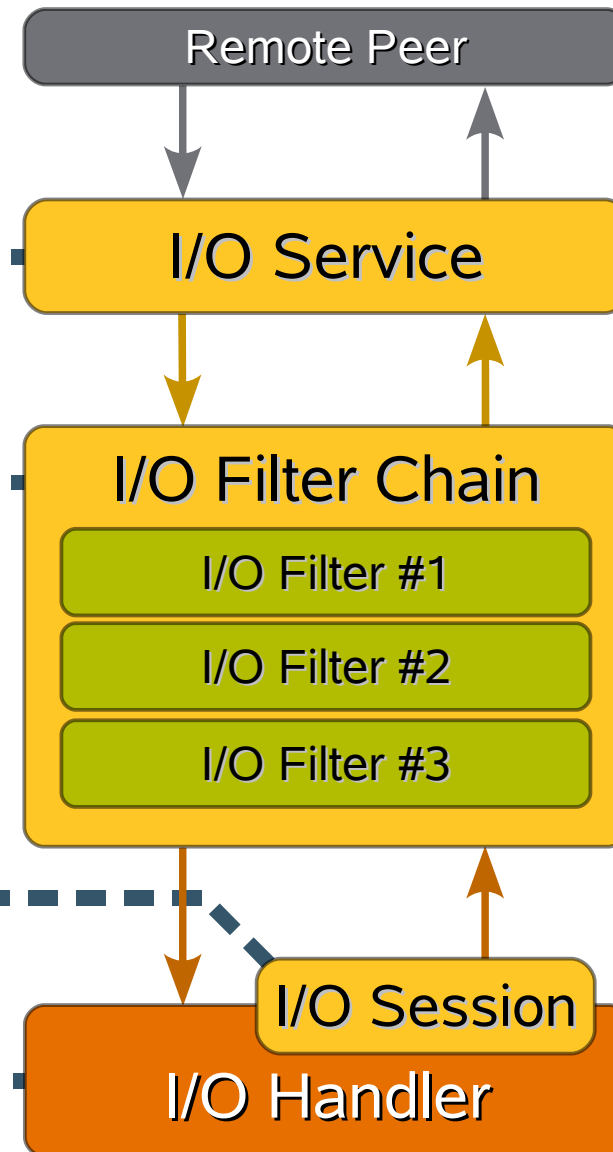
What does it look like?

➤ 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 = ...;  
  
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.

```
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.

```
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.

```
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 easy like this.

```
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

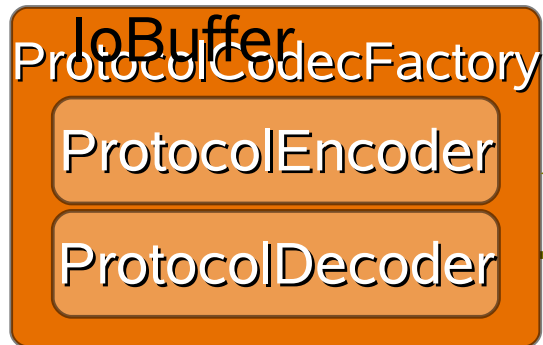
Why do we need a protocol codec?

- 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)

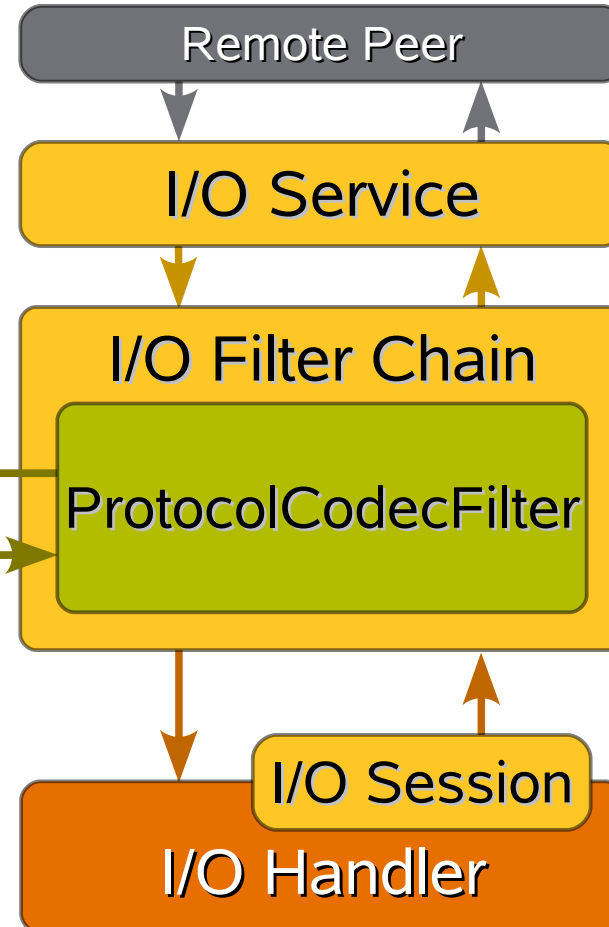
Protocol Codecs

What does it look like with a protocol codec?

➤ POJO →



➤ IoBuffer →
POJO



Protocol Codecs

Echo server redux – TextLineProtocolCodecFactory
kicks in!

```
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!?

```
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.

```
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
 - Traffic throttling
 - Changing thread model

- ▶ JMIImplementation
- ▶ com.sun.management
- ▶ java.lang
- ▶ java.util.logging
- ▼ org.apache.mina
 - ▶ filter
 - ▼ service
 - ▼  myService
 - ▶ **Attributes**
 - ▶ Operations
 - ▶ Notifications

Attribute values

Name	Value
activationTime	Fri Apr 04 21:03:45 KST 2008
active	true
backlog	50
bothIdle	false
bothIdleCount	0
bothIdleTime	0
bothIdleTimeInMillis	0
closeOnDeactivation	true
cumulativeManagedSessionCount	1
defaultLocalAddresses	*:0
disposed	false
disposing	false
filterChainBuilder	{codec=org.apache.mina.filt...
handler	JmxTest\$1
largestManagedSessionCount	1
largestReadBytesThroughput	2.6657780739753414
largestReadMessagesThroughput	0.3333333333333333
largestWrittenBytesThroughput	2.332555814728424
largestWrittenMessagesThroughput	0.3333333333333333

- ▶ JMIImplementation
- ▶ com.sun.management
- ▶ java.lang
- ▶ java.util.logging
- ▼ org.apache.mina
 - ▶ filter
 - ▼ service
 - ▼  myService
 - ▶ Attributes
 - ▶ **Operations**
 - ▶ Notifications

Operation invocation

void **bind** ()

void **bind** (p1)

void **unbind** ()

void **unbind** (p1)

void **dispose** ()

java.util.Set **findSessions** (ognlQuery)



java.util.Set **findAndRegisterSessions** (ognlQuery)

java.util.Set **findAndProcessSessions** (ognlQuery)

- ▶ JMIImplementation
- ▶ com.sun.management
- ▶ java.lang
- ▶ java.util.logging
- ▼ org.apache.mina
 - ▶ filter
 - ▼ service
 - ▼  myService
 - ▶ Attributes
 - ▶ Operations
 - ▶ Notifications
 - ▼ session
 - ▼  0x1338933D
 - ▶ **Attributes**
 - ▶ Operations
 - ▶ Notifications

Attribute values

Name	Value
bothIdle	false
bothIdleCount	0
closing	false
config.bothIdleTime	0
config.bothIdleTimeInMillis	0
config.keepAlive	false
config.maxReadBufferSize	65536
config.minReadBufferSize	64
config.oobInline	false
config.readBufferSize	128
config.readerIdleTime	0
config.readerIdleTimeInMillis	0
config.receiveBufferSize	43744
config.reuseAddress	true
config.sendBufferSize	25146
config.soLinger	-1
config.tcpNoDelay	false
config.throughputCalculationInte...	3
config.throughputCalculationInte...	3000

- ▶ JMIImplementation
- ▶ com.sun.management
- ▶ java.lang
- ▶ java.util.logging
- ▼ org.apache.mina
 - ▶ filter
 - ▼ service
 - ▼  myService
 - ▶ Attributes
 - ▶ Operations
 - ▶ Notifications
 - ▼ session
 - ▼  0x1338933D
 - ▶ Attributes
 - ▶ **Operations**
 - ▶ Notifications

Operation invocation

java.lang.String **close** ()

java.lang.String **closeOnFlush** ()

void **suspendRead** ()

void **suspendWrite** ()

void **resumeRead** ()

void **resumeWrite** ()

void **addFilterFirst** (name , filter {

void **addFilterLast** (name , filter {

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.

THANK YOU

Trustin Lee
Principal Software Engineer
Red Hat, Inc.

TS-4814

