Developing Pluggable Client/Server Applications with Net4j
Agenda

- Requirements
- Architecture
  - Buffers
  - Channels
  - Connectors
  - Acceptors
  - Protocols
  - Signals
- Examples
- Discussion
Requirements

- High performance
  - java.nio.DirectByteBuffer, zero copying, non-blocking
- Good scalability
  - java.nio.channels.Selector, single I/O thread possible
- Multiple transports
  - Shipped with TCP and JVM transports
- Pluggable protocols
  - Independent of chosen transport
- Server-initiated push services (agent paradigm)
  - Asynchronous and synchronous requests from the server
- OSGi and stand-alone modes
Buffers

IBufferHandler

IBuffer

BufferOutputStream

BufferInputStream

BufferState

IBufferPool

extends

IBufferHandler handles BufferState

IBufferProvider

reads

fetches buffers

writes

passes buffers
Connectors

- ConnectorLocation
- ConnectorState
- IChannelMultiplexer
- IConnector
- TCPConnector
- JVMConnector
- IChannel
- IProtocol

- ConnectorLocation extends IConnector
- ConnectorState implements IConnector
- IChannelMultiplexer extends IChannel
- IConnector implements TCPConnector, JVMConnector
- IChannel creates IProtocol
- IProtocol configures ConnectorState
Acceptors

JVMAcceptor implements IAcceptor

JVMConnector implements IConnector

TCPAcceptor implements IAcceptor

TCPConnector implements IConnector
Signals

- SignalProtocol
  - implements IProtocol
  - creates Signal
    - extends SignalActor
      - extends Request
      - extends RequestWithConfirmation
    - extends SignalReactor
      - extends Indication
      - extends IndicationWithResponse
      - runs in Thread

Developing Pluggable Client/Server Applications with Net4j | © 2008 by Eike Stepper; made available under the EPL v1.0
public class JMSLogonRequest extends RequestWithConfirmation<Boolean> {
    private String userName;
    private String password;

    public JMSLogonRequest(IChannel channel, String userName, String password) {
        super(channel);
        this.userName = userName;
        this.password = password;
    }

    @Override
    protected short getSignalID() { return JMSProtocolConstants.SIGNAL_LOGON; }

    @Override
    protected void requesting(ExtendedDataOutputStream out) throws IOException {
        out.writeString(userName);
        out.writeString(password);
    }

    @Override
    protected Boolean confirming(ExtendedDataInputStream in) throws IOException {
        return in.readBoolean();
    }
}
public class JMSLogonIndication extends IndicationWithResponse {

    private boolean ok;

    @Override
    protected short getSignalID() {
        return JMSProtocolConstants.SIGNAL_LOGON;
    }

    @Override
    protected void indicating(ExtendedDataInputStream in) throws IOException {
        String userName = in.readString();
        String password = in.readString();
        ok = JMSServer.INSTANCE.logon(userName, password);
    }

    @Override
    protected void responding(ExtendedDataOutputStream out) throws IOException {
        out.writeBoolean(ok);
    }
}
public class JMSServerProtocol extends SignalProtocol
{
    public String getType()
    {
        return JMSProtocolConstants.PROTOCOL_NAME;
    }

    @Override
    protected SignalReactor doCreateSignalReactor(short signalID)
    {
        switch (signalID)
        {
            case JMSProtocolConstants.SIGNAL_SYNC:
                return new JMSSyncIndication();

            case JMSProtocolConstants.SIGNAL_LOGON:
                return new JMSLogonIndication();

            default:
                return null;
        }
    }
}
// Start a TCP acceptor that is configured through extension points
IAcceptor acceptor = TCPUtil.getAcceptor(IPluginContainer.INSTANCE, "0.0.0:2036");

// Open a TCP connection that is configured through extension points
IConnector connector = TCPUtil.getConnector(IPluginContainer.INSTANCE, "localhost:2036");

// Open a channel with the JMS protocol
IChannel channel = connector.openChannel(JMSProtocolConstants.PROTOCOL_NAME);

try
{
    // Create a logon request and send it through the channel
    JMSLogonRequest request = new JMSLogonRequest(channel, "stepper", "secret");
    boolean ok = request.send();
}
catch (Exception ex)
{
    OM.LOG.error("Problem during logon", ex);
}
finally
{
    channel.close();
}
Thank you for listening!
http://wiki.eclipse.org/Net4j
http://wiki.eclipse.org/CDO

Questions?
Comments?
Suggestions?