The Net4j Signalling Platform

Developing Pluggable Client/Server Applications
Agenda

• Requirements
• Architecture
  • Buffers
  • Channels
  • Connectors
  • Acceptors
  • Protocols
  • Signals
• Examples
• FileShare Demo
Requirements

• High performance
  - java.nio.DirectByteBuffer, zero copying

• Good scalability
  - java.nio.channels.Selector, single I/O thread possible

• Multiple transports
  - Shipped with TCP, HTTP and JVM transports

• Pluggable protocols
  - Independent of chosen transport

• Server-initiated push services (agent paradigm)
  - Asynchronous and synchronous requests towards the client

• OSGi™ and stand-alone modes
Architecture

- TCP
- JVM
- App1
- App2
- Acceptors
- Signals
- Connectors
- Protocols
- Channels
- Buffers
- Utils
- OSGi / Eclipse
Connectors

- **ConnectorLocation**
- **ConnectorState**
- **IChannelMultiplexer**
- **IConnector**
- **TCPConnector**
- **JVMConnector**
- **IBufferHandler**
- **IChannel**

**Relationships**
- **extends** IConnector
- **creates** IBufferHandler
- **implements** IChannelMultiplexer
- **implements** IChannel
- **extends** ConnectorLocation
- **extends** ConnectorState

**Diagram Notes**
- Developed with Pluggable Client/Server Applications with Net4J | © 2009 by Eike Stepper; made available under the EPL v1.0
Acceptors

- JVMAcceptor
  - creates
  - implements
- IAcceptor
  - implements
- TCPAcceptor
  - creates

- JVMConnector
  - implements
- IConnector
  - implements
- TCPConnector
  - implements
Protocols

- IBufferHandler
- IProtocol
- IChannel
- IProtocolProvider

- IProtocol extends IInterface
- IProtocol uses IInterface
- IChannel provides client protocol
- IProtocolProvider provides server protocol
Signals

- SignalProtocol creates Signal
- SignalProtocol implements IProtocol
- Signal extends SignalActor and SignalReactor
- Signal runs in Thread

- SignalActor extends Request and RequestWithConfirmation
- SignalReactor extends Indication and IndicationWithResponse

Developing Pluggable Client/Server Applications with Net4j | © 2009 by Eike Stepper; made available under the EPL v1.0
```java
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 JMServerProtocol 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();
        }

        return null;
    }
}
// Start a TCP acceptor that is configured through extension points
IAcceptor acceptor = TCPUtil.getAcceptor(IPPluginContainer.INSTANCE, "0.0.0.0:2036");

// Open a TCP connection that is configured through extension points
IConnector connector = TCPUtil.getConnector(IPPluginContainer.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();
}
FileShare Demo