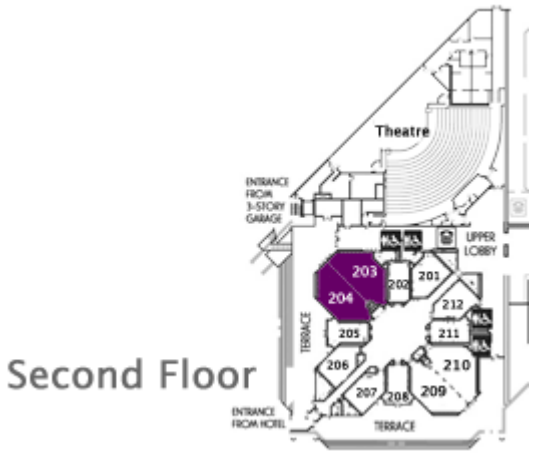
eclipseCON™ 2008

Developing Pluggable Client/Server Applications with Net4j



Thursday, 13:40, 10 minutes | Room 203/204 |

7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 · 15 · 16 · 17 · 18 · 19

© 2008 by Eike Stepper; made available under the EPL v1.0 | 03-20-2008

Agenda

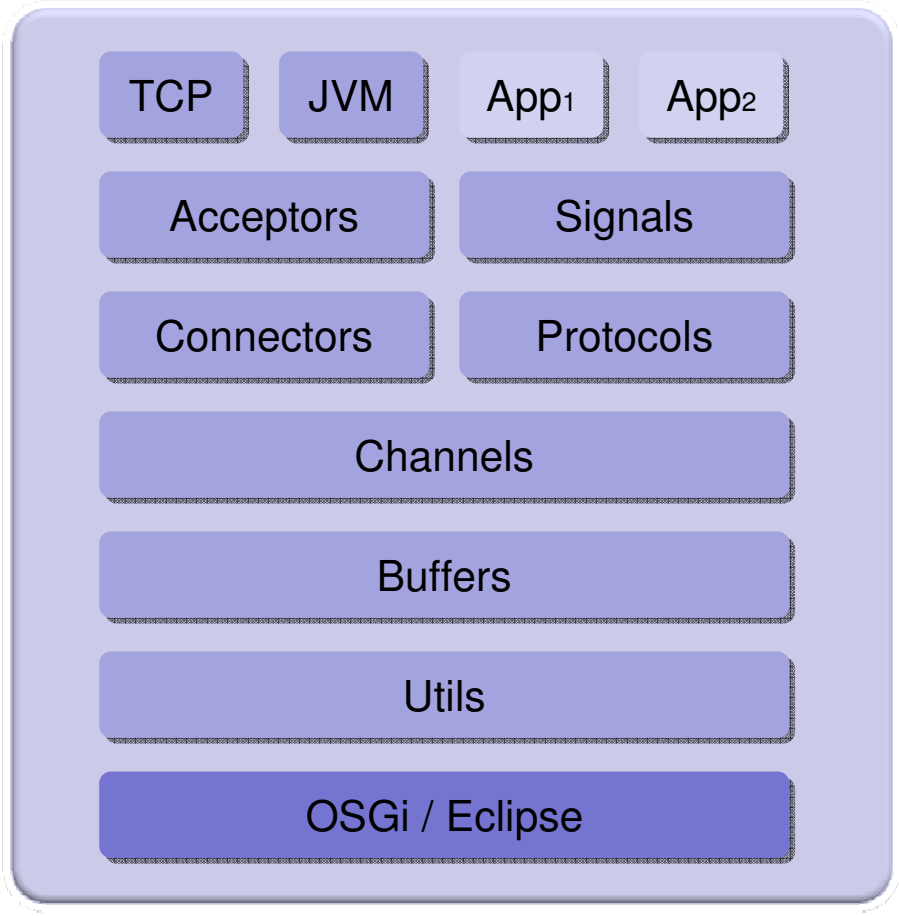
- Requirements
- Architecture
  - Buffers
  - Channels
  - Connectors
  - Acceptors
  - Protocols
  - Signals
- Examples
- Discussion

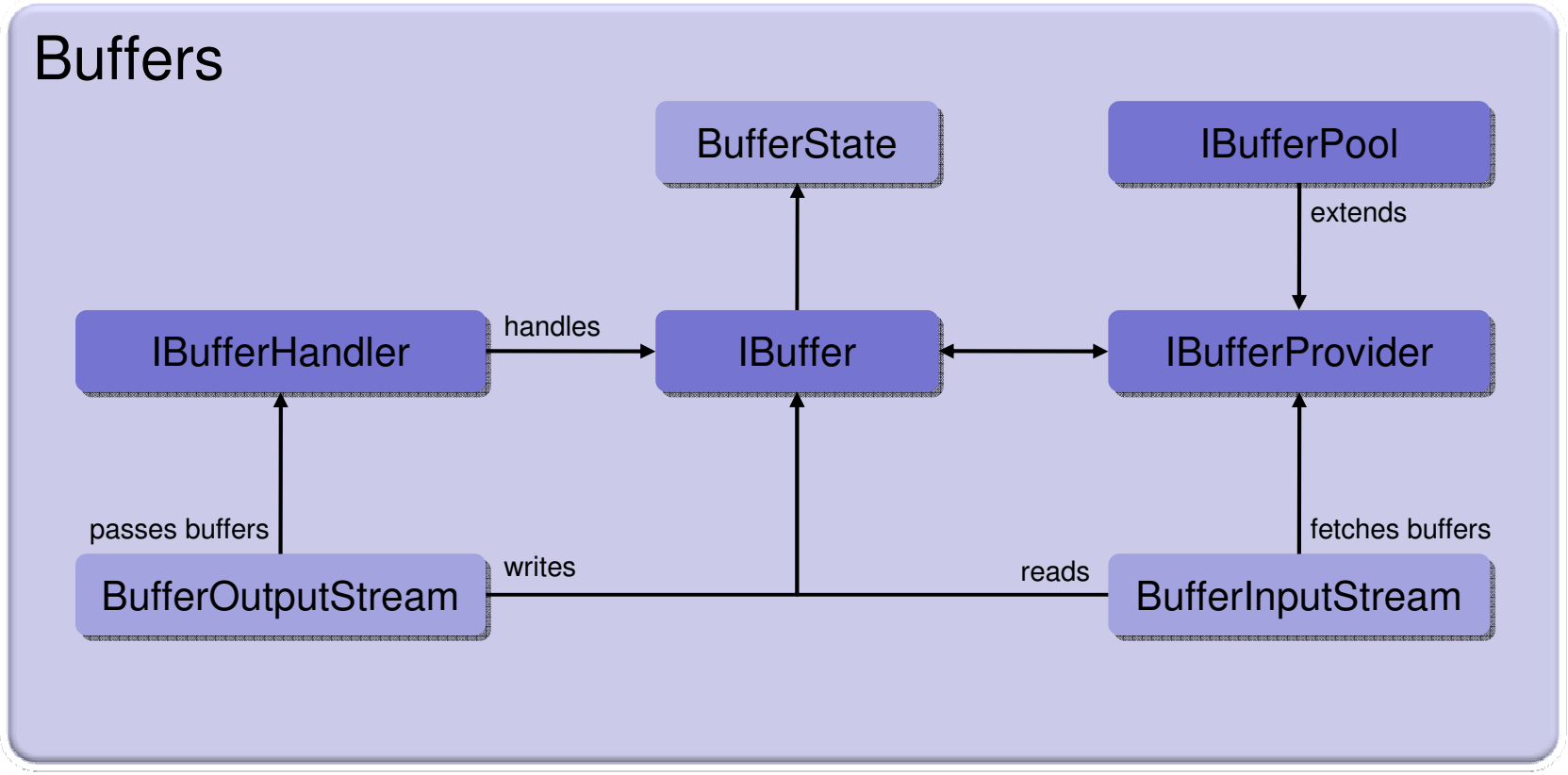
 | © 2008 by Eike Stepper; made available under the EPL v1.0

Requirements

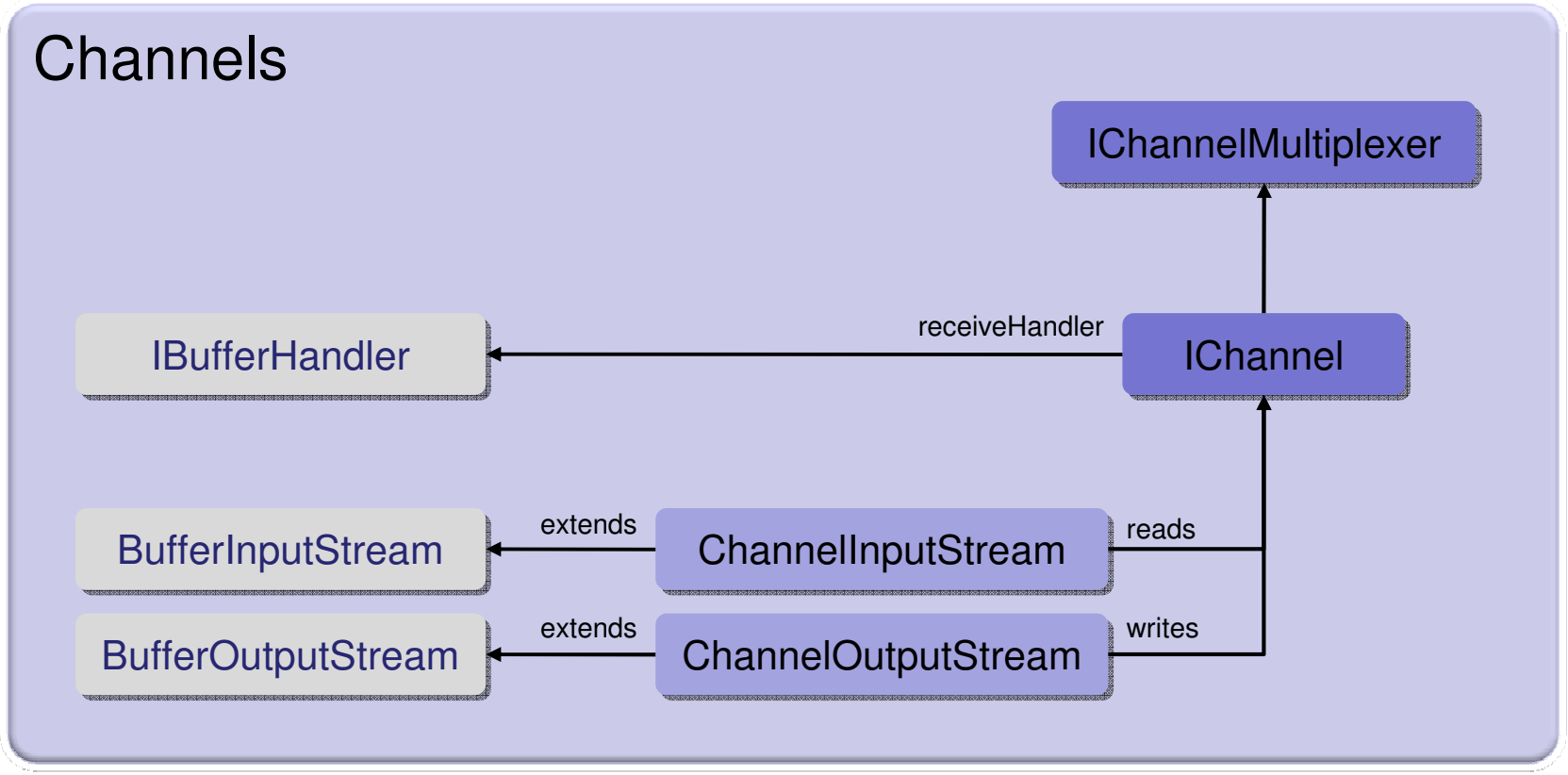
- High performance
  - java.nio.DirectByteBuffer, zero copying, non-blocking
- Good scalability
  - java.nio.channels.Selector, single I/O thread possible
- Multiple transports
  - Shipped with TCP and JVM transports
- Pluggable protocols
  - Independent of chosen transport
- Server-initiated push services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

Architecture



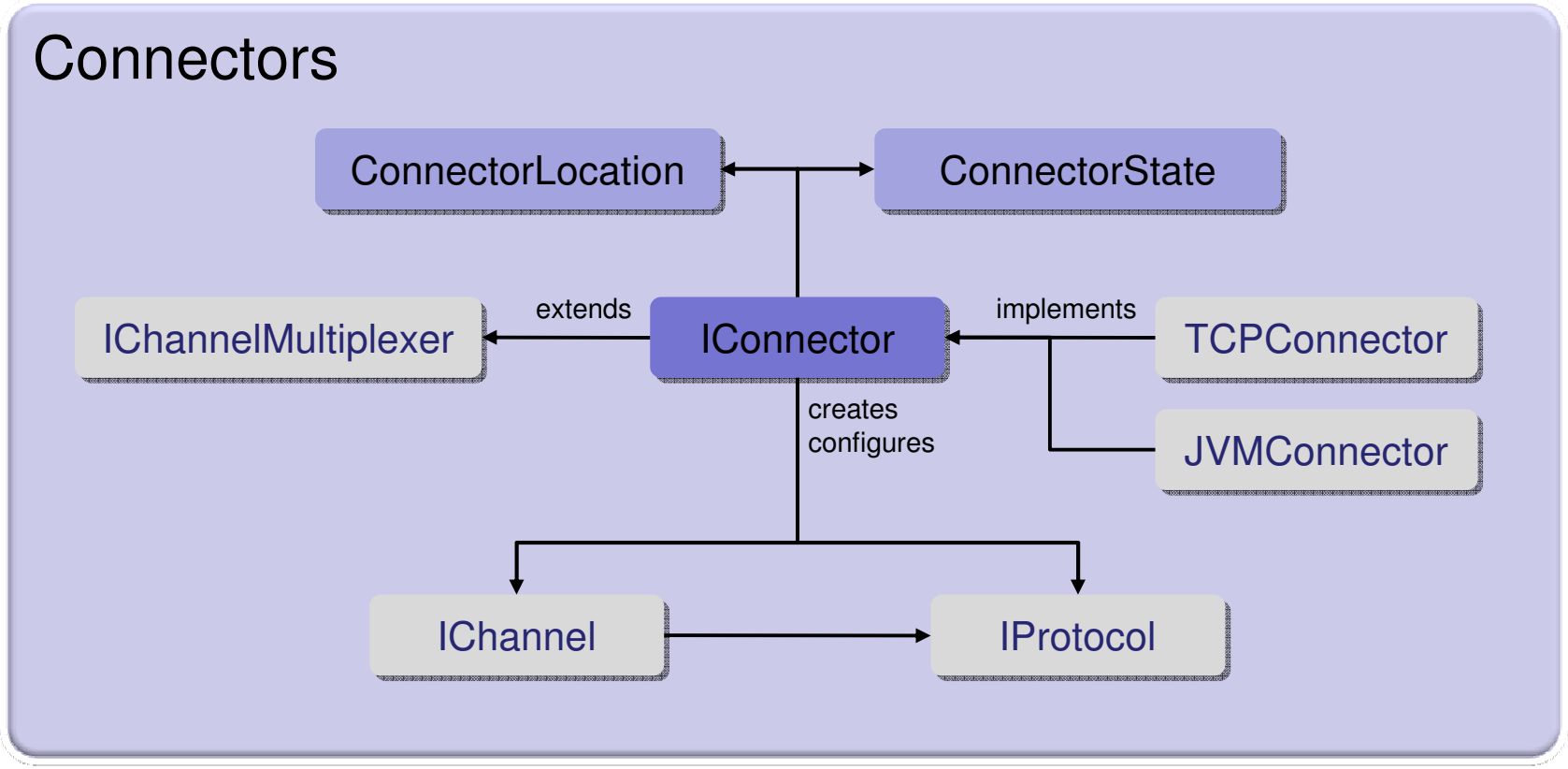

Buffers
BufferState
IBufferPool
extends
IBufferHandler
handles
IBuffer
IBufferProvider
passes buffers
writes
reads
fetches buffers
BufferOutputStream
BufferInputStream

Channels

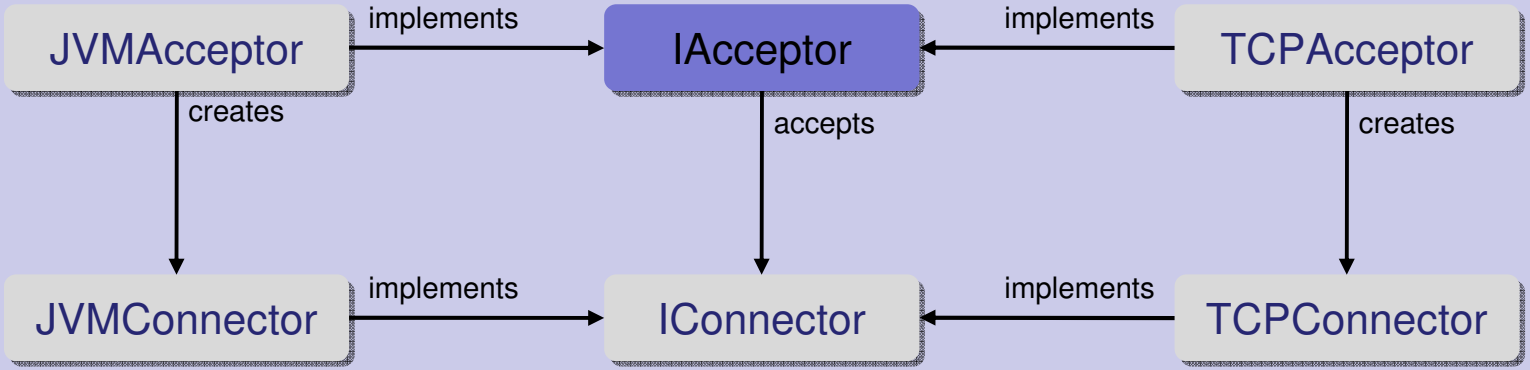


 | © 2008 by Eike Stepper; made available under the EPL v1.0

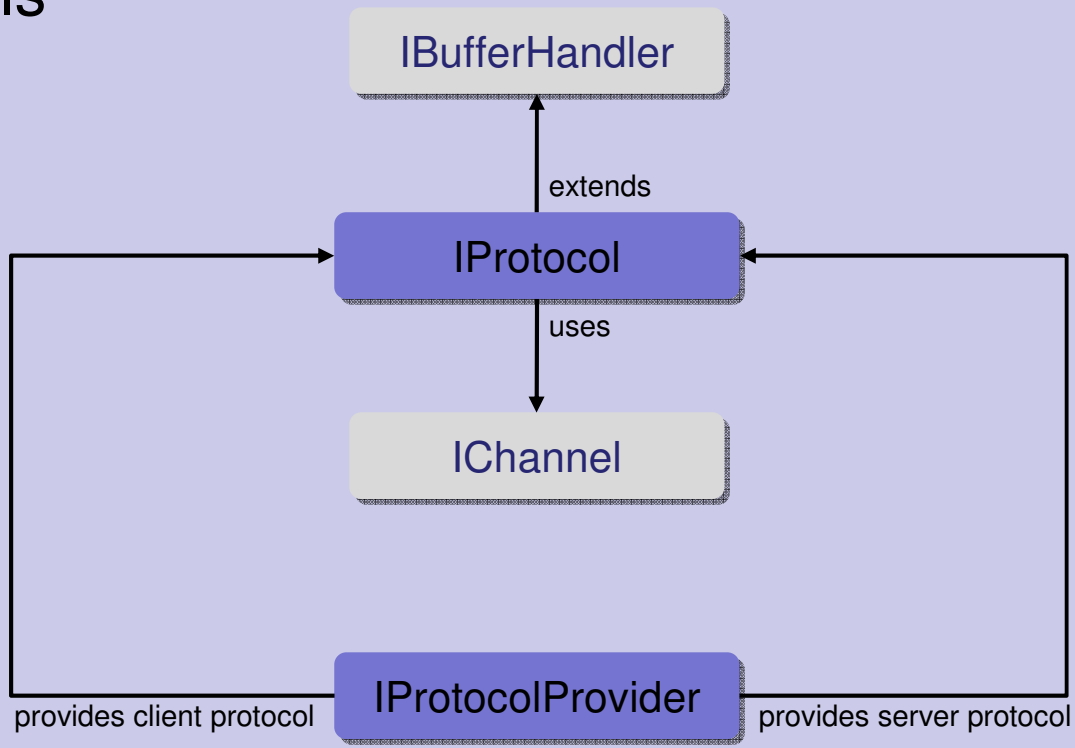
Connectors


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

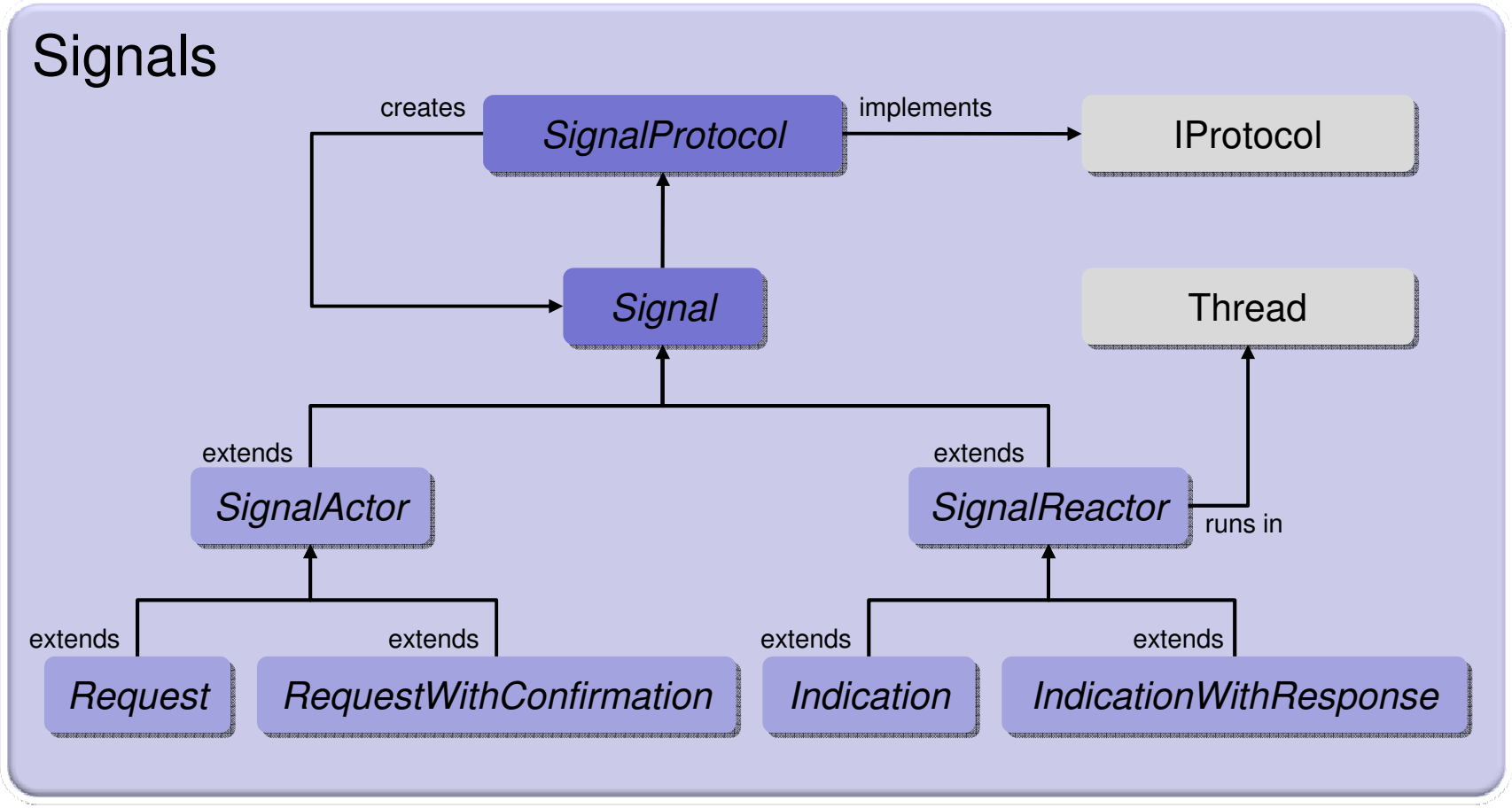
Acceptors


JVMAcceptor
implements
IAcceptor
implements
TCPAcceptor
creates
accepts
creates
JVMConnector
implements
IConnector
implements
TCPConnector

Protocols


IBufferHandler
extends
IProtocol
uses
IChannel
IProtocolProvider
provides client protocol
provides server protocol

Signals


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

Examples

```
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();
  }
}
```

Examples

```
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);
  }
}
```

Examples

```
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;
    }
  }
}
```

Examples

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

// 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();
}
```

Discussion

Thank you for listening!

http://wiki.eclipse.org/Net4j

http://wiki.eclipse.org/CDO

Questions?

Comments?

Suggestions?