第十二章：SPDY

目录 [-]

本章我将不会直接翻译Netty In Action书中的原文，感觉原书中本章讲的很多废话，我翻译起来也吃力。所以，本章内容我会根据其他资料和个人理解来讲述。

12.1 SPDY概念及背景

SPDY 是 Google 开发的基于传输控制协议 (TCP) 的应用层协议 ，开发组正在推动 SPDY 成为正式标准（现为互联网草案）。SPDY 协议旨在通过压缩、多路复用和优先级来缩短网页的加载时间和提高安全性。（SPDY 是 Speedy 的昵音，意思是更快）。

为什么需要SPDY？SPDY 协议只是在性能上对 HTTP 做了很大的优化，其核心思想是尽量减少连接个数，而对于 HTTP 的语义并没有做太大的修改。具体来说是，SPDY 使用了 HTTP 的方法和页眉，但是删除了一些头并重写了 HTTP 中管理连接和数据转移格式的部分，所以基本上是兼容 HTTP 的。

Google 在 SPDY 白皮书里表示要向协议栈下面渗透并替换掉传输层协议（TCP），但是因为这样无论是部署起来还是实现起来暂时相当困难，因此 Google 准备先对应用层协议 HTTP 进行改进，先在 SSL 之上增加一个会话层来实现 SPDY 协议，而 HTTP 的 GET 和 POST 消息格式保持不变，即现有的所有服务端应用均不用做任何修改。因此在目前，SPDY 的目的是为了加强 HTTP，是对 HTTP 一个更好的实现和支持。至于未来 SPDY 得到广泛应用后会不会演一出狸猫换太子，替换掉 HTTP 并彻底颠覆整个 Internet 就是 Google 的事情了。

距离万维网之父蒂姆·伯纳斯-李发明并推动 HTTP 成为如今互联网最流行的协议已经过去十几年了（现用 HTTP 1.1 规范也停滞了 13 年了），随着现在 WEB 技术的飞速发展尤其是 HTML5 的不断演进，包括 WebSockets 协议的出现以及当前网络环境的改变、传输内容的变化，当初的 HTTP 规范已经逐渐无法满足人们的需要了，HTTP 需要进一步发展，因此 HTTPbis 工作组已经被组建并被授权考虑 HTTP 2.0 ，希望能解决掉目前 HTTP 所带来的诸多限制。而 SPDY 正是 Google 在 HTTP 即将从 1.1 跨越到 2.0 之际推出的试图成为下一代互联网通信的协议，长期以来一直被认为是 HTTP 2.0 唯一可行选择。

SPDY相比HTTP有如下优点：

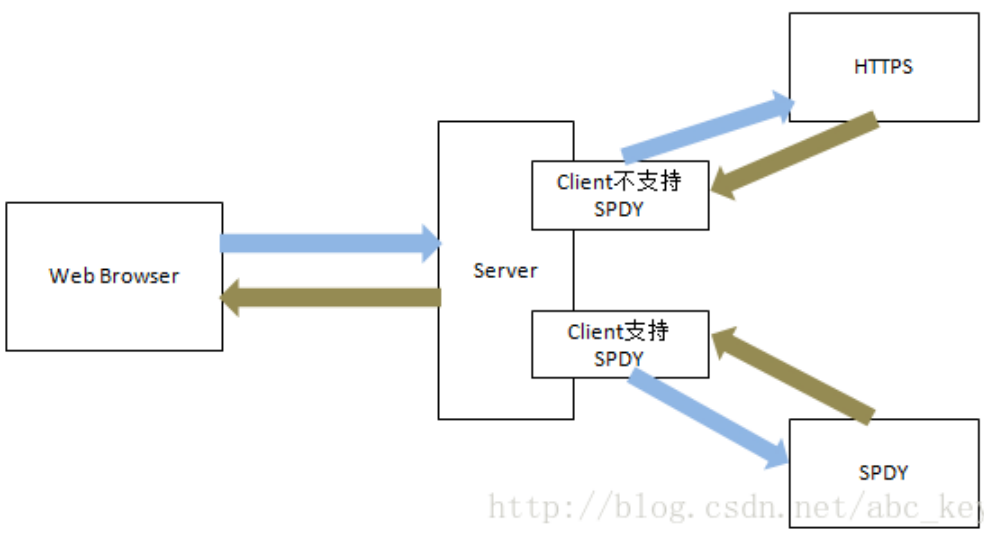
1. SPDY多路复用，请求优化；而HTTP单路连接，请求低效
2. SPDY支持服务器推送技术；而HTTP只允许由客户端主动发起请求
3. SPDY压缩了HTTP头信息，节省了传输数据的带宽流量；而HTTP头冗余，同一个会话会反复送头信息
4. SPDY强制使用SSL传输协议，全部请求SSL加密后，信息传输更安全

谷歌表示，引入SPDY协议后，在实验室测试中页面加载速度比原先快64%。

支持SPDY协议的浏览器：

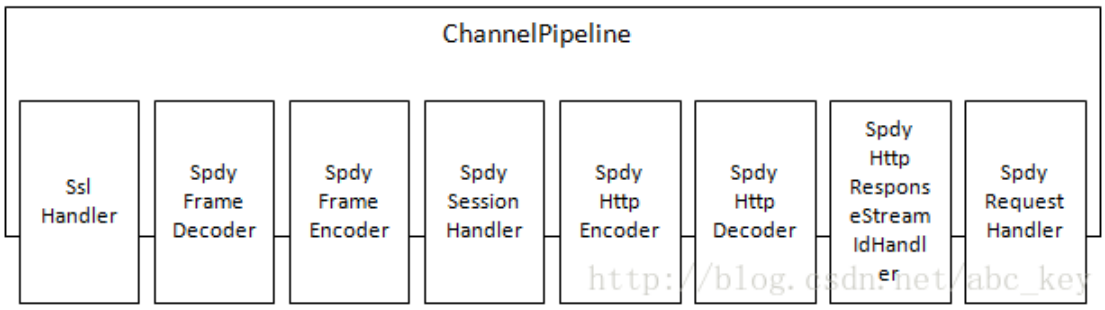
- Google Chrome 19+和Chromium 19+
- Mozilla Firefox 11+，从13开始默认支持
- Opera 12.10+
- Internet Explorer 11+

12.2 本例子流程图

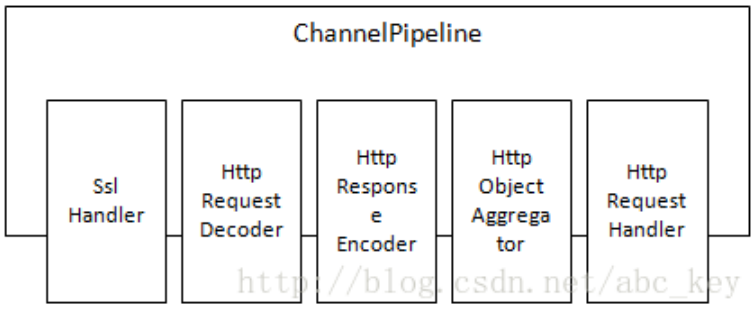


12.3 Netty中使用SPDY

支持SPDY的ChannelPipeline如下图：



不支持SPDY的ChannelPipeline如下图：



例子代码如下：

```java
package netty.in.action.spdy;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import org.eclipse.jetty.npn.NextProtoNego.ServerProvider;

public class DefaultServerProvider implements ServerProvider {

    private static final List<String> PROTOCOLS = Collections.unmodifiableList(Arrays
            .asList("spdy/3.1", "http/1.1", "http/1.0", "Unknown"));

    private String protocol;

    public String getSelectedProtocol() {
        return protocol;
    }

    @Override
    public void protocolSelected(String arg0) {
        this.protocol = arg0;
    }

    @Override
    public List<String> protocols() {
        return PROTOCOLS;
    }

    @Override
    public void unsupported() {
        protocol = "http/1.1";
    }

}
```

```java
package netty.in.action.spdy;

import io.netty.channel.ChannelFuture;
import io.netty.channel.ChannelFutureListener;
import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.SimpleChannelInboundHandler;
import io.netty.handler.codec.http.DefaultFullHttpResponse;
import io.netty.handler.codec.http.FullHttpRequest;
import io.netty.handler.codec.http.FullHttpResponse;
import io.netty.handler.codec.http.HttpHeaders;
import io.netty.handler.codec.http.HttpResponseStatus;
import io.netty.handler.codec.http.HttpVersion;
import io.netty.util.CharsetUtil;

public class HttpRequestHandler extends SimpleChannelInboundHandler<FullHttpRequest> {

```

```
    @Override
    protected void channelRead0(ChannelHandlerContext ctx, FullHttpRequest request)
            throws Exception {
        if (HttpHeaders.is100ContinueExpected(request)) {
            send100Continue(ctx);
        }
        FullHttpResponse response = new DefaultFullHttpResponse(
                request.getProtocolVersion(), HttpResponseStatus.OK);
        response.content().writeBytes(getContent().getBytes(CharsetUtil.UTF_8));
        response.headers().set(HttpHeaders.Names.CONTENT_TYPE,
                "text/plain; charset=UTF-8");
        boolean keepAlive = HttpHeaders.isKeepAlive(request);
        if (keepAlive) {
            response.headers().set(HttpHeaders.Names.CONTENT_LENGTH,
                    response.content().readableBytes());
            response.headers().set(HttpHeaders.Names.CONNECTION,
                    HttpHeaders.Values.KEEP_ALIVE);
        }
        ChannelFuture future = ctx.writeAndFlush(response);
        if (!keepAlive) {
            future.addListener(ChannelFutureListener.CLOSE);
        }
    }

    private static void send100Continue(ChannelHandlerContext ctx) {
        FullHttpResponse response = new DefaultFullHttpResponse(HttpVersion.HTTP_1_1,
                HttpResponseStatus.CONTINUE);
        ctx.writeAndFlush(response);
    }

    protected String getContent() {
        return "This content is transmitted via HTTP\r\n";
    }

    @Override
    public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause)
            throws Exception {
        cause.printStackTrace();
        ctx.close();
    }
}
```

[java]

```
package netty.in.action.spdy;

public class SpdyRequestHandler extends HttpRequestHandler {

    @Override
    protected String getContent() {
        return "This content is transmitted via SPDY\r\n";
    }

}
```

[java]

```
package netty.in.action.spdy;

import io.netty.channel.ChannelInboundHandler;
import io.netty.handler.codec.spdy.SpdyOrHttpChooser;

import javax.net.ssl.SSLEngine;

import org.eclipse.jetty.npn.NextProtoNego;

public class DefaultSpdyOrHttpChooser extends SpdyOrHttpChooser {

    protected DefaultSpdyOrHttpChooser(int maxSpdyContentLength, int maxHttpContentLength) {
        super(maxSpdyContentLength, maxHttpContentLength);
    }

    @Override
    protected SelectedProtocol getProtocol(SSLEngine engine) {
        DefaultServerProvider provider = (DefaultServerProvider) NextProtoNego
                .get(engine);
        String protocol = provider.getSelectedProtocol();
        if (protocol == null) {
            return SelectedProtocol.UNKNOWN;
```

```
        }
        switch (protocol) {
        case "spdy/3.1":
            return SelectedProtocol.SPDY_3_1;
        case "http/1.0":
        case "http/1.1":
            return SelectedProtocol.HTTP_1_1;
        default:
            return SelectedProtocol.UNKNOWN;
        }
    }

    @Override
    protected ChannelInboundHandler createHttpRequestHandlerForHttp() {
        return new HttpRequestHandler();
    }

    @Override
    protected ChannelInboundHandler createHttpRequestHandlerForSpdy() {
        return new SpdyRequestHandler();
    }

}
```

[java]

```
package netty.in.action.spdy;

import io.netty.channel.Channel;
import io.netty.channel.ChannelInitializer;
import io.netty.channel.ChannelPipeline;
import io.netty.handler.ssl.SslHandler;

import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLEngine;

import org.eclipse.jetty.npn.NextProtoNego;

public class SpdyChannelInitializer extends ChannelInitializer<Channel> {
    private final SSLContext context;

    public SpdyChannelInitializer(SSLContext context) {
        this.context = context;
    }

    @Override
    protected void initChannel(Channel ch) throws Exception {
        ChannelPipeline pipeline = ch.pipeline();
        SSLEngine engine = context.createSSLEngine();
        engine.setUseClientMode(false);
        NextProtoNego.put(engine, new DefaultServerProvider());
        NextProtoNego.debug = true;
        pipeline.addLast("sslHandler", new SslHandler(engine));
        pipeline.addLast("chooser",
                new DefaultSpdyOrHttpChooser(1024 * 1024, 1024 * 1024));
    }

}
```

[java]

```
package netty.in.action.spdy;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.Channel;
import io.netty.channel.ChannelFuture;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.example.securechat.SecureChatSslContextFactory;

import java.net.InetSocketAddress;

import javax.net.ssl.SSLContext;

public class SpdyServer {

    private final NioEventLoopGroup group = new NioEventLoopGroup();
    private final SSLContext context;
    private Channel channel;
```

```
19.
20.     public SpdyServer(SSLContext context) {
21.         this.context = context;
22.     }
23.
24.     public ChannelFuture start(InetSocketAddress address) {
25.         ServerBootstrap bootstrap = new ServerBootstrap();
26.         bootstrap.group(group).channel(NioServerSocketChannel.class)
27.                 .childHandler(new SpdyChannelInitializer(context));
28.         ChannelFuture future = bootstrap.bind(address);
29.         future.syncUninterruptibly();
30.         channel = future.channel();
31.         return future;
32.     }
33.
34.     public void destroy() {
35.         if (channel != null) {
36.             channel.close();
37.         }
38.         group.shutdownGracefully();
39.     }
40.
41.     public static void main(String[] args) {
42.         SSLContext context = SecureChatSslContextFactory.getServerContext();
43.         final SpdyServer endpoint = new SpdyServer(context);
44.         ChannelFuture future = endpoint.start(new InetSocketAddress(4096));
45.         Runtime.getRuntime().addShutdownHook(new Thread() {
46.             @Override
47.             public void run() {
48.                 endpoint.destroy();
49.             }
50.         });
51.         future.channel().closeFuture().syncUninterruptibly();
52.     }
53.
54. }
```

使用SSL需要使用到SSLContext，下面代买是获取SSLContext对象：

```
[java]
01. /*
02.  * Copyright 2012 The Netty Project
03.  *
04.  * The Netty Project licenses this file to you under the Apache License,
05.  * version 2.0 (the "License"); you may not use this file except in compliance
06.  * with the License. You may obtain a copy of the License at:
07.  *
08.  *   http://www.apache.org/licenses/LICENSE-2.0
09.  *
10.  * Unless required by applicable law or agreed to in writing, software
11.  * distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
12.  * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
13.  * License for the specific language governing permissions and limitations
14.  * under the License.
15.  */
16. package netty.in.action.spdy;
17.
18. import javax.net.ssl.ManagerFactoryParameters;
19. import javax.net.ssl.TrustManager;
20. import javax.net.ssl.TrustManagerFactorySpi;
21. import javax.net.ssl.X509TrustManager;
22. import java.security.InvalidAlgorithmParameterException;
23. import java.security.KeyStore;
24. import java.security.KeyStoreException;
25. import java.security.cert.X509Certificate;
26.
27. /**
28.  * Bogus {@link TrustManagerFactorySpi} which accepts any certificate
29.  * even if it is invalid.
30.  */
31. public class SecureChatTrustManagerFactory extends TrustManagerFactorySpi {
32.
33.     private static final TrustManager DUMMY_TRUST_MANAGER = new X509TrustManager() {
34.         @Override
35.         public X509Certificate[] getAcceptedIssuers() {
36.             return new X509Certificate[0];
37.         }
38.
```

```
        @Override
        public void checkClientTrusted(X509Certificate[] chain, String authType) {
            // Always trust - it is an example.
            // You should do something in the real world.
            // You will reach here only if you enabled client certificate auth,
            // as described in SecureChatSslContextFactory.
            System.err.println(
                    "UNKNOWN CLIENT CERTIFICATE: " + chain[0].getSubjectDN());
        }

        @Override
        public void checkServerTrusted(X509Certificate[] chain, String authType) {
            // Always trust - it is an example.
            // You should do something in the real world.
            System.err.println(
                    "UNKNOWN SERVER CERTIFICATE: " + chain[0].getSubjectDN());
        }
    };

    public static TrustManager[] getTrustManagers() {
        return new TrustManager[] { DUMMY_TRUST_MANAGER };
    }

    @Override
    protected TrustManager[] engineGetTrustManagers() {
        return getTrustManagers();
    }

    @Override
    protected void engineInit(KeyStore keystore) throws KeyStoreException {
        // Unused
    }

    @Override
    protected void engineInit(ManagerFactoryParameters managerFactoryParameters)
            throws InvalidAlgorithmParameterException {
        // Unused
    }
}
```

[java]

```
/*
 * Copyright 2012 The Netty Project
 *
 * The Netty Project licenses this file to you under the Apache License,
 * version 2.0 (the "License"); you may not use this file except in compliance
 * with the License. You may obtain a copy of the License at:
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
 * License for the specific language governing permissions and limitations
 * under the License.
 */
package netty.in.action.spdy;

import java.io.ByteArrayInputStream;
import java.io.InputStream;

/**
 * A bogus key store which provides all the required information to
 * create an example SSL connection.
 *
 * To generate a bogus key store:
 * <pre>
 * keytool  -genkey -alias securechat -keysize 2048 -validity 36500
 *          -keyalg RSA -dname "CN=securechat"
 *          -keypass secret -storepass secret
 *          -keystore cert.jks
 * </pre>
 */
public final class SecureChatKeyStore {
    private static final short[] DATA = {
        0xfe, 0xed, 0xfe, 0xed, 0x00, 0x00, 0x00, 0x02,
        0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01,
        0x00, 0x07, 0x65, 0x78, 0x61, 0x6d, 0x70, 0x6c,
```

```
        0x65, 0x00, 0x00, 0x01, 0x1a, 0x9f, 0x57, 0xa5,
        0x27, 0x00, 0x00, 0x01, 0x9a, 0x30, 0x82, 0x01,
        0x96, 0x30, 0x0e, 0x06, 0x0a, 0x2b, 0x06, 0x01,
        0x04, 0x01, 0x2a, 0x02, 0x11, 0x01, 0x01, 0x05,
        0x00, 0x04, 0x82, 0x01, 0x82, 0x48, 0x6d, 0xcf,
        0x16, 0xb5, 0x50, 0x95, 0x36, 0xbf, 0x47, 0x27,
        0x50, 0x58, 0x0d, 0xa2, 0x52, 0x7e, 0x25, 0xab,
        0x14, 0x1a, 0x26, 0x5e, 0x2d, 0x8a, 0x23, 0x90,
        0x60, 0x7f, 0x12, 0x20, 0x56, 0xd1, 0x43, 0xa2,
        0x6b, 0x47, 0x5d, 0xed, 0x9d, 0xd4, 0xe5, 0x83,
        0x28, 0x89, 0xc2, 0x16, 0x4c, 0x76, 0x06, 0xad,
        0x8e, 0x8c, 0x29, 0x1a, 0x9b, 0x0f, 0xdd, 0x60,
        0x4b, 0xb4, 0x62, 0x82, 0x9e, 0x4a, 0x63, 0x83,
        0x2e, 0xd2, 0x43, 0x78, 0xc2, 0x32, 0x1f, 0x60,
        0xa9, 0x8a, 0x7f, 0x0f, 0x7c, 0xa6, 0x1d, 0xe6,
        0x92, 0x9e, 0x52, 0xc7, 0x7d, 0xbb, 0x35, 0x3b,
        0xaa, 0x89, 0x73, 0x4c, 0xfb, 0x99, 0x54, 0x97,
        0x99, 0x28, 0x6e, 0x66, 0x5b, 0xf7, 0x9b, 0x7e,
        0x6d, 0x8a, 0x2f, 0xfa, 0xc3, 0x1e, 0x71, 0xb9,
        0xbd, 0x8f, 0xc5, 0x63, 0x25, 0x31, 0x20, 0x02,
        0xff, 0x02, 0xf0, 0xc9, 0x2c, 0xdd, 0x3a, 0x10,
        0x30, 0xab, 0xe5, 0xad, 0x3d, 0x1a, 0x82, 0x77,
        0x46, 0xed, 0x03, 0x38, 0xa4, 0x73, 0x6d, 0x36,
        0x36, 0x33, 0x70, 0xb2, 0x63, 0x20, 0xca, 0x03,
        0xbf, 0x5a, 0xf4, 0x7c, 0x35, 0xf0, 0x63, 0x1a,
        0x12, 0x33, 0x12, 0x58, 0xd9, 0xa2, 0x63, 0x6b,
        0x63, 0x82, 0x41, 0x65, 0x70, 0x37, 0x4b, 0x99,
        0x04, 0x9f, 0xdd, 0x5e, 0x07, 0x01, 0x95, 0x9f,
        0x36, 0xe8, 0xc3, 0x66, 0x2a, 0x21, 0x69, 0x68,
        0x40, 0xe6, 0xbc, 0xbb, 0x85, 0x81, 0x21, 0x13,
        0xe6, 0xa4, 0xcf, 0xd3, 0x67, 0xe3, 0xfd, 0x75,
        0xf0, 0xdf, 0x83, 0xe0, 0xc5, 0x36, 0x09, 0xac,
        0x1b, 0xd4, 0xf7, 0x2a, 0x23, 0x57, 0x1c, 0x5c,
        0x0f, 0xf4, 0xcf, 0xa2, 0xcf, 0xf5, 0xbd, 0x9c,
        0x69, 0x98, 0x78, 0x3a, 0x25, 0xe4, 0xfd, 0x85,
        0x11, 0xcc, 0x7d, 0xef, 0xeb, 0x74, 0x60, 0xb1,
        0xb7, 0xfb, 0x1f, 0x0e, 0x62, 0xff, 0xfe, 0x09,
        0x0a, 0xc3, 0x80, 0x2f, 0x10, 0x49, 0x89, 0x78,
        0xd2, 0x08, 0xfa, 0x89, 0x22, 0x45, 0x91, 0x21,
        0xbc, 0x90, 0x3e, 0xad, 0xb3, 0x0a, 0xb4, 0x0e,
        0x1c, 0xa1, 0x93, 0x92, 0xd8, 0x72, 0x07, 0x54,
        0x60, 0xe7, 0x91, 0xfc, 0xd9, 0x3c, 0xe1, 0x6f,
        0x08, 0xe4, 0x56, 0xf6, 0x0b, 0xb0, 0x3c, 0x39,
        0x8a, 0x2d, 0x48, 0x44, 0x28, 0x13, 0xca, 0xe9,
        0xf7, 0xa3, 0xb6, 0x8a, 0x5f, 0x31, 0xa9, 0x72,
        0xf2, 0xde, 0x96, 0xf2, 0xb1, 0x53, 0xb1, 0x3e,
        0x24, 0x57, 0xfd, 0x18, 0x45, 0x1f, 0xc5, 0x33,
        0x1b, 0xa4, 0xe8, 0x21, 0xfa, 0x0e, 0xb2, 0xb9,
        0xcb, 0xc7, 0x07, 0x41, 0xdd, 0x2f, 0xb6, 0x6a,
        0x23, 0x18, 0xed, 0xc1, 0xef, 0xe2, 0x4b, 0xec,
        0xc9, 0xba, 0xfb, 0x46, 0x43, 0x90, 0xd7, 0xb5,
        0x68, 0x28, 0x31, 0x2b, 0x8d, 0xa8, 0x51, 0x63,
        0xf7, 0x53, 0x99, 0x19, 0x68, 0x85, 0x66, 0x00,
        0x00, 0x00, 0x01, 0x00, 0x05, 0x58, 0x2e, 0x35,
        0x30, 0x39, 0x00, 0x00, 0x02, 0x3a, 0x30, 0x82,
        0x02, 0x36, 0x30, 0x82, 0x01, 0xe0, 0xa0, 0x03,
        0x02, 0x01, 0x02, 0x02, 0x04, 0x48, 0x59, 0xf1,
        0x92, 0x30, 0x0d, 0x06, 0x09, 0x2a, 0x86, 0x48,
        0x86, 0xf7, 0x0d, 0x01, 0x01, 0x05, 0x05, 0x00,
        0x30, 0x81, 0xa0, 0x31, 0x0b, 0x30, 0x09, 0x06,
        0x03, 0x55, 0x04, 0x06, 0x13, 0x02, 0x4b, 0x52,
        0x31, 0x13, 0x30, 0x11, 0x06, 0x03, 0x55, 0x04,
        0x08, 0x13, 0x0a, 0x4b, 0x79, 0x75, 0x6e, 0x67,
        0x67, 0x69, 0x2d, 0x64, 0x6f, 0x31, 0x14, 0x30,
        0x12, 0x06, 0x03, 0x55, 0x04, 0x07, 0x13, 0x0b,
        0x53, 0x65, 0x6f, 0x6e, 0x67, 0x6e, 0x61, 0x6d,
        0x2d, 0x73, 0x69, 0x31, 0x1a, 0x30, 0x18, 0x06,
        0x03, 0x55, 0x04, 0x0a, 0x13, 0x11, 0x54, 0x68,
        0x65, 0x20, 0x4e, 0x65, 0x74, 0x74, 0x79, 0x20,
        0x50, 0x72, 0x6f, 0x6a, 0x65, 0x63, 0x74, 0x31,
        0x18, 0x30, 0x16, 0x06, 0x03, 0x55, 0x04, 0x0b,
        0x13, 0x0f, 0x45, 0x78, 0x61, 0x6d, 0x70, 0x6c,
        0x65, 0x20, 0x41, 0x75, 0x74, 0x68, 0x6f, 0x72,
        0x73, 0x31, 0x30, 0x30, 0x2e, 0x06, 0x03, 0x55,
        0x04, 0x03, 0x13, 0x27, 0x73, 0x65, 0x63, 0x75,
        0x72, 0x65, 0x63, 0x68, 0x61, 0x74, 0x2e, 0x65,
        0x78, 0x61, 0x6d, 0x70, 0x6c, 0x65, 0x2e, 0x6e,
        0x65, 0x74, 0x74, 0x79, 0x2e, 0x67, 0x6c, 0x65,
        0x61, 0x6d, 0x79, 0x6e, 0x6f, 0x64, 0x65, 0x2e,
```

```
        0x6e, 0x65, 0x74, 0x30, 0x20, 0x17, 0x0d, 0x30,
        0x38, 0x30, 0x36, 0x31, 0x39, 0x30, 0x35, 0x34,
        0x31, 0x33, 0x38, 0x5a, 0x18, 0x0f, 0x32, 0x31,
        0x38, 0x37, 0x31, 0x31, 0x32, 0x34, 0x30, 0x35,
        0x34, 0x31, 0x33, 0x38, 0x5a, 0x30, 0x81, 0xa0,
        0x31, 0x0b, 0x30, 0x09, 0x06, 0x03, 0x55, 0x04,
        0x06, 0x13, 0x02, 0x4b, 0x52, 0x31, 0x13, 0x30,
        0x11, 0x06, 0x03, 0x55, 0x04, 0x08, 0x13, 0x0a,
        0x4b, 0x79, 0x75, 0x6e, 0x67, 0x67, 0x69, 0x2d,
        0x64, 0x6f, 0x31, 0x14, 0x30, 0x12, 0x06, 0x03,
        0x55, 0x04, 0x07, 0x13, 0x0b, 0x53, 0x65, 0x6f,
        0x6e, 0x67, 0x6e, 0x61, 0x6d, 0x2d, 0x73, 0x69,
        0x31, 0x1a, 0x30, 0x18, 0x06, 0x03, 0x55, 0x04,
        0x0a, 0x13, 0x11, 0x54, 0x68, 0x65, 0x20, 0x4e,
        0x65, 0x74, 0x74, 0x79, 0x20, 0x50, 0x72, 0x6f,
        0x6a, 0x65, 0x63, 0x74, 0x31, 0x18, 0x30, 0x16,
        0x06, 0x03, 0x55, 0x04, 0x0b, 0x13, 0x0f, 0x45,
        0x78, 0x61, 0x6d, 0x70, 0x6c, 0x65, 0x20, 0x41,
        0x75, 0x74, 0x68, 0x6f, 0x72, 0x73, 0x31, 0x30,
        0x30, 0x2e, 0x06, 0x03, 0x55, 0x04, 0x03, 0x13,
        0x27, 0x73, 0x65, 0x63, 0x75, 0x72, 0x65, 0x63,
        0x68, 0x61, 0x74, 0x2e, 0x65, 0x78, 0x61, 0x6d,
        0x70, 0x6c, 0x65, 0x2e, 0x6e, 0x65, 0x74, 0x74,
        0x79, 0x2e, 0x67, 0x6c, 0x65, 0x61, 0x6d, 0x79,
        0x6e, 0x6f, 0x64, 0x65, 0x2e, 0x6e, 0x65, 0x74,
        0x30, 0x5c, 0x30, 0x0d, 0x06, 0x09, 0x2a, 0x86,
        0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x01, 0x05,
        0x00, 0x03, 0x4b, 0x00, 0x30, 0x48, 0x02, 0x41,
        0x00, 0xc3, 0xe3, 0x5e, 0x41, 0xa7, 0x87, 0x11,
        0x00, 0x42, 0x2a, 0xb0, 0x4b, 0xed, 0xb2, 0xe0,
        0x23, 0xdb, 0xb1, 0x3d, 0x58, 0x97, 0x35, 0x60,
        0x0b, 0x82, 0x59, 0xd3, 0x00, 0xea, 0xd4, 0x61,
        0xb8, 0x79, 0x3f, 0xb6, 0x3c, 0x12, 0x05, 0x93,
        0x2e, 0x9a, 0x59, 0x68, 0x14, 0x77, 0x3a, 0xc8,
        0x50, 0x25, 0x57, 0xa4, 0x49, 0x18, 0x63, 0x41,
        0xf0, 0x2d, 0x28, 0xec, 0x06, 0xfb, 0xb4, 0x9f,
        0xbf, 0x02, 0x03, 0x01, 0x00, 0x01, 0x30, 0x0d,
        0x06, 0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d,
        0x01, 0x01, 0x05, 0x05, 0x00, 0x03, 0x41, 0x00,
        0x65, 0x6c, 0x30, 0x01, 0xc2, 0x8e, 0x3e, 0xcb,
        0xb3, 0x77, 0x48, 0xe9, 0x66, 0x61, 0x9a, 0x40,
        0x86, 0xaf, 0xf6, 0x03, 0xeb, 0xba, 0x6a, 0xf2,
        0xfd, 0xe2, 0xaf, 0x36, 0x5e, 0x7b, 0xaa, 0x22,
        0x04, 0xdd, 0x2c, 0x20, 0xc4, 0xfc, 0xdd, 0xd0,
        0x82, 0x20, 0x1c, 0x3d, 0xd7, 0x9e, 0x5e, 0x5c,
        0x92, 0x5a, 0x76, 0x71, 0x28, 0xf5, 0x07, 0x7d,
        0xa2, 0x81, 0xba, 0x77, 0x9f, 0x2a, 0xd9, 0x44,
        0x00, 0x00, 0x00, 0x01, 0x00, 0x05, 0x6d, 0x79,
        0x6b, 0x65, 0x79, 0x00, 0x00, 0x01, 0x1a, 0x9f,
        0x5b, 0x56, 0xa0, 0x00, 0x00, 0x01, 0x99, 0x30,
        0x82, 0x01, 0x95, 0x30, 0x0e, 0x06, 0x0a, 0x2b,
        0x06, 0x01, 0x04, 0x01, 0x2a, 0x02, 0x11, 0x01,
        0x01, 0x05, 0x00, 0x04, 0x82, 0x01, 0x81, 0x29,
        0xa8, 0xb6, 0x08, 0x0c, 0x85, 0x75, 0x3e, 0xdd,
        0xb5, 0xe5, 0x1a, 0x87, 0x68, 0xd1, 0x90, 0x4b,
        0x29, 0x31, 0xee, 0x90, 0xbc, 0x9d, 0x73, 0xa0,
        0x3f, 0xe9, 0x0b, 0xa4, 0xef, 0x30, 0x9b, 0x36,
        0x9a, 0xb2, 0x54, 0x77, 0x81, 0x07, 0x4b, 0xaa,
        0xa5, 0x77, 0x98, 0xe1, 0xeb, 0xb5, 0x7c, 0x4e,
        0x48, 0xd5, 0x08, 0xfc, 0x2c, 0x36, 0xe2, 0x65,
        0x03, 0xac, 0xe5, 0xf3, 0x96, 0xb7, 0xd0, 0xb5,
        0x3b, 0x92, 0xe4, 0x14, 0x05, 0x7a, 0x6a, 0x92,
        0x56, 0xfe, 0x4e, 0xab, 0xd3, 0x0e, 0x32, 0x04,
        0x22, 0x22, 0x74, 0x47, 0x7d, 0xec, 0x21, 0x99,
        0x30, 0x31, 0x64, 0x46, 0x64, 0x9b, 0xc7, 0x13,
        0xbf, 0xbe, 0xd0, 0x31, 0x49, 0xe7, 0x3c, 0xbf,
        0xba, 0xb1, 0x20, 0xf9, 0x42, 0xf4, 0xa9, 0xa9,
        0xe5, 0x13, 0x65, 0x32, 0xbf, 0x7c, 0xcc, 0x91,
        0xd3, 0xfd, 0x24, 0x47, 0x0b, 0xe5, 0x53, 0xad,
        0x50, 0x30, 0x56, 0xd1, 0xfa, 0x9c, 0x37, 0xa8,
        0xc1, 0xce, 0xf6, 0x0b, 0x18, 0xaa, 0x7c, 0xab,
        0xbd, 0x1f, 0xdf, 0xe4, 0x80, 0xb8, 0xa7, 0xe0,
        0xad, 0x7d, 0x50, 0x74, 0xf1, 0x98, 0x78, 0xbc,
        0x58, 0xb9, 0xc2, 0x52, 0xbe, 0xd2, 0x5b, 0x81,
        0x94, 0x83, 0x8f, 0xb9, 0x4c, 0xee, 0x01, 0x2b,
        0x5e, 0xc9, 0x6e, 0x9b, 0xf5, 0x63, 0x69, 0xe4,
        0xd8, 0x0b, 0x47, 0xd8, 0xfd, 0xd8, 0xe0, 0xed,
        0xa8, 0x27, 0x03, 0x74, 0x1e, 0x5d, 0x32, 0xe6,
        0x5c, 0x63, 0xc2, 0xfb, 0x3f, 0xee, 0xb4, 0x13,
```

```
        0xc6, 0x0e, 0x6e, 0x74, 0xe0, 0x22, 0xac, 0xce,
        0x79, 0xf9, 0x43, 0x68, 0xc1, 0x03, 0x74, 0x2b,
        0xe1, 0x18, 0xf8, 0x7f, 0x76, 0x9a, 0xea, 0x82,
        0x3f, 0xc2, 0xa6, 0xa7, 0x4c, 0xfe, 0xae, 0x29,
        0x3b, 0xc1, 0x10, 0x7c, 0xd5, 0x77, 0x17, 0x79,
        0x5f, 0xcb, 0xad, 0x1f, 0xd8, 0xa1, 0xfd, 0x90,
        0xe1, 0x6b, 0xb2, 0xef, 0xb9, 0x41, 0x26, 0xa4,
        0x0b, 0x4f, 0xc6, 0x83, 0x05, 0x6f, 0xf0, 0x64,
        0x40, 0xe1, 0x44, 0xc4, 0xf9, 0x40, 0x2b, 0x3b,
        0x40, 0xdb, 0xaf, 0x35, 0xa4, 0x9b, 0x9f, 0xc4,
        0x74, 0x07, 0xe5, 0x18, 0x60, 0xc5, 0xfe, 0x15,
        0x0e, 0x3a, 0x25, 0x2a, 0x11, 0xee, 0x78, 0x2f,
        0xb8, 0xd1, 0x6e, 0x4e, 0x3c, 0x0a, 0xb5, 0xb9,
        0x40, 0x86, 0x27, 0x6d, 0x8f, 0x53, 0xb7, 0x77,
        0x36, 0xec, 0x5d, 0xed, 0x32, 0x40, 0x43, 0x82,
        0xc3, 0x52, 0x58, 0xc4, 0x26, 0x39, 0xf3, 0xb3,
        0xad, 0x58, 0xab, 0xb7, 0xf7, 0x8e, 0x0e, 0xba,
        0x8e, 0x78, 0x9d, 0xbf, 0x58, 0x34, 0xbd, 0x77,
        0x73, 0xa6, 0x50, 0x55, 0x00, 0x60, 0x26, 0xbf,
        0x6d, 0xb4, 0x98, 0x8a, 0x18, 0x83, 0x89, 0xf8,
        0xcd, 0x0d, 0x49, 0x06, 0xae, 0x51, 0x6e, 0xaf,
        0xbd, 0xe2, 0x07, 0x13, 0xd8, 0x64, 0xcc, 0xbf,
        0x00, 0x00, 0x00, 0x01, 0x00, 0x05, 0x58, 0x2e,
        0x35, 0x30, 0x39, 0x00, 0x00, 0x02, 0x34, 0x30,
        0x82, 0x02, 0x30, 0x30, 0x82, 0x01, 0xda, 0xa0,
        0x03, 0x02, 0x01, 0x02, 0x02, 0x04, 0x48, 0x59,
        0xf2, 0x84, 0x30, 0x0d, 0x06, 0x09, 0x2a, 0x86,
        0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x05, 0x05,
        0x00, 0x30, 0x81, 0x9d, 0x31, 0x0b, 0x30, 0x09,
        0x06, 0x03, 0x55, 0x04, 0x06, 0x13, 0x02, 0x4b,
        0x52, 0x31, 0x13, 0x30, 0x11, 0x06, 0x03, 0x55,
        0x04, 0x08, 0x13, 0x0a, 0x4b, 0x79, 0x75, 0x6e,
        0x67, 0x67, 0x69, 0x2d, 0x64, 0x6f, 0x31, 0x14,
        0x30, 0x12, 0x06, 0x03, 0x55, 0x04, 0x07, 0x13,
        0x0b, 0x53, 0x65, 0x6f, 0x6e, 0x67, 0x6e, 0x61,
        0x6d, 0x2d, 0x73, 0x69, 0x31, 0x1a, 0x30, 0x18,
        0x06, 0x03, 0x55, 0x04, 0x0a, 0x13, 0x11, 0x54,
        0x68, 0x65, 0x20, 0x4e, 0x65, 0x74, 0x74, 0x79,
        0x20, 0x50, 0x72, 0x6f, 0x6a, 0x65, 0x63, 0x74,
        0x31, 0x15, 0x30, 0x13, 0x06, 0x03, 0x55, 0x04,
        0x0b, 0x13, 0x0c, 0x43, 0x6f, 0x6e, 0x74, 0x72,
        0x69, 0x62, 0x75, 0x74, 0x6f, 0x72, 0x73, 0x31,
        0x30, 0x30, 0x2e, 0x06, 0x03, 0x55, 0x04, 0x03,
        0x13, 0x27, 0x73, 0x65, 0x63, 0x75, 0x72, 0x65,
        0x63, 0x68, 0x61, 0x74, 0x2e, 0x65, 0x78, 0x61,
        0x6d, 0x70, 0x6c, 0x65, 0x2e, 0x6e, 0x65, 0x74,
        0x74, 0x79, 0x2e, 0x67, 0x6c, 0x65, 0x61, 0x6d,
        0x79, 0x6e, 0x6f, 0x64, 0x65, 0x2e, 0x6e, 0x65,
        0x74, 0x30, 0x20, 0x17, 0x0d, 0x30, 0x38, 0x30,
        0x36, 0x31, 0x39, 0x30, 0x35, 0x34, 0x35, 0x34,
        0x30, 0x5a, 0x18, 0x0f, 0x32, 0x31, 0x38, 0x37,
        0x31, 0x31, 0x32, 0x33, 0x30, 0x35, 0x34, 0x35,
        0x34, 0x30, 0x5a, 0x30, 0x81, 0x9d, 0x31, 0x0b,
        0x30, 0x09, 0x06, 0x03, 0x55, 0x04, 0x06, 0x13,
        0x02, 0x4b, 0x52, 0x31, 0x13, 0x30, 0x11, 0x06,
        0x03, 0x55, 0x04, 0x08, 0x13, 0x0a, 0x4b, 0x79,
        0x75, 0x6e, 0x67, 0x67, 0x69, 0x2d, 0x64, 0x6f,
        0x31, 0x14, 0x30, 0x12, 0x06, 0x03, 0x55, 0x04,
        0x07, 0x13, 0x0b, 0x53, 0x65, 0x6f, 0x6e, 0x67,
        0x6e, 0x61, 0x6d, 0x2d, 0x73, 0x69, 0x31, 0x1a,
        0x30, 0x18, 0x06, 0x03, 0x55, 0x04, 0x0a, 0x13,
        0x11, 0x54, 0x68, 0x65, 0x20, 0x4e, 0x65, 0x74,
        0x74, 0x79, 0x20, 0x50, 0x72, 0x6f, 0x6a, 0x65,
        0x63, 0x74, 0x31, 0x15, 0x30, 0x13, 0x06, 0x03,
        0x55, 0x04, 0x0b, 0x13, 0x0c, 0x43, 0x6f, 0x6e,
        0x74, 0x72, 0x69, 0x62, 0x75, 0x74, 0x6f, 0x72,
        0x73, 0x31, 0x30, 0x30, 0x2e, 0x06, 0x03, 0x55,
        0x04, 0x03, 0x13, 0x27, 0x73, 0x65, 0x63, 0x75,
        0x72, 0x65, 0x63, 0x68, 0x61, 0x74, 0x2e, 0x65,
        0x78, 0x61, 0x6d, 0x70, 0x6c, 0x65, 0x2e, 0x6e,
        0x65, 0x74, 0x74, 0x79, 0x2e, 0x67, 0x6c, 0x65,
        0x61, 0x6d, 0x79, 0x6e, 0x6f, 0x64, 0x65, 0x2e,
        0x6e, 0x65, 0x74, 0x30, 0x5c, 0x30, 0x0d, 0x06,
        0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d, 0x01,
        0x01, 0x01, 0x05, 0x00, 0x03, 0x4b, 0x00, 0x30,
        0x48, 0x02, 0x41, 0x00, 0x95, 0xb3, 0x47, 0x17,
        0x95, 0x0f, 0x57, 0xcf, 0x66, 0x72, 0x0a, 0x7e,
        0x5b, 0x54, 0xea, 0x8c, 0x6f, 0x79, 0xde, 0x94,
        0xac, 0x0b, 0x5a, 0xd4, 0xd6, 0x1b, 0x58, 0x12,
```

```
        0x1a, 0x16, 0x3d, 0xfe, 0xdf, 0xa5, 0x2b, 0x86,
        0xbc, 0x64, 0xd4, 0x80, 0x1e, 0x3f, 0xf9, 0xe2,
        0x04, 0x03, 0x79, 0x9b, 0xc1, 0x5c, 0xf0, 0xf1,
        0xf3, 0xf1, 0xe3, 0xbf, 0x3f, 0xc0, 0x1f, 0xdd,
        0xdb, 0xc0, 0x5b, 0x21, 0x02, 0x03, 0x01, 0x00,
        0x01, 0x30, 0x0d, 0x06, 0x09, 0x2a, 0x86, 0x48,
        0x86, 0xf7, 0x0d, 0x01, 0x01, 0x05, 0x05, 0x00,
        0x03, 0x41, 0x00, 0x02, 0xd7, 0xdd, 0xbd, 0x0c,
        0x8e, 0x21, 0x20, 0xef, 0x9e, 0x4f, 0x1f, 0xf5,
        0x49, 0xf1, 0xae, 0x58, 0x9b, 0x94, 0x3a, 0x1f,
        0x70, 0x33, 0xf0, 0x9b, 0xbb, 0xe9, 0xc0, 0xf3,
        0x72, 0xcb, 0xde, 0xb6, 0x56, 0x72, 0xcc, 0x1c,
        0xf0, 0xd6, 0x5a, 0x2a, 0xbc, 0xa1, 0x7e, 0x23,
        0x83, 0xe9, 0xe7, 0xcf, 0x9e, 0xa5, 0xf9, 0xcc,
        0xc2, 0x61, 0xf4, 0xdb, 0x40, 0x93, 0x1d, 0x63,
        0x8a, 0x50, 0x4c, 0x11, 0x39, 0xb1, 0x91, 0xc1,
        0xe6, 0x9d, 0xd9, 0x1a, 0x62, 0x1b, 0xb8, 0xd3,
        0xd6, 0x9a, 0x6d, 0xb9, 0x8e, 0x15, 0x51 };

    public static InputStream asInputStream() {
        byte[] data = new byte[DATA.length];
        for (int i = 0; i < data.length; i ++) {
            data[i] = (byte) DATA[i];
        }
        return new ByteArrayInputStream(data);
    }

    public static char[] getCertificatePassword() {
        return "secret".toCharArray();
    }

    public static char[] getKeyStorePassword() {
        return "secret".toCharArray();
    }

    private SecureChatKeyStore() {
        // Unused
    }
}
```

[java]

```java
/*
 * Copyright 2012 The Netty Project
 *
 * The Netty Project licenses this file to you under the Apache License,
 * version 2.0 (the "License"); you may not use this file except in compliance
 * with the License. You may obtain a copy of the License at:
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
 * License for the specific language governing permissions and limitations
 * under the License.
 */
package netty.in.action.spdy;

import io.netty.handler.ssl.SslHandler;
import io.netty.util.internal.SystemPropertyUtil;

import java.security.KeyStore;
import java.security.SecureRandom;

import javax.net.ssl.KeyManager;
import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLEngine;
import javax.net.ssl.TrustManager;

/**
 * Creates a bogus {@link SSLContext}.  A client-side context created by this
 * factory accepts any certificate even if it is invalid.  A server-side context
 * created by this factory sends a bogus certificate defined in {@link SecureChatKeyStore}.
 * <p>
 * You will have to create your context differently in a real world application.
 *
 * <h3>Client Certificate Authentication</h3>
```

```
 *
 * To enable client certificate authentication:
 * <ul>
 * <li>Enable client authentication on the server side by calling
 *     {@link SSLEngine#setNeedClientAuth(boolean)} before creating
 *     {@link SslHandler}.</li>
 * <li>When initializing an {@link SSLContext} on the client side,
 *     specify the {@link KeyManager} that contains the client certificate as
 *     the first argument of {@link SSLContext#init(KeyManager[], TrustManager[], SecureRandom)}.</li>
 * <li>When initializing an {@link SSLContext} on the server side,
 *     specify the proper {@link TrustManager} as the second argument of
 *     {@link SSLContext#init(KeyManager[], TrustManager[], SecureRandom)}
 *     to validate the client certificate.</li>
 * </ul>
 */
public final class SecureChatSslContextFactory {

    private static final String PROTOCOL = "TLS";
    private static final SSLContext SERVER_CONTEXT;
    private static final SSLContext CLIENT_CONTEXT;

    static {
        String algorithm = SystemPropertyUtil.get("ssl.KeyManagerFactory.algorithm");
        if (algorithm == null) {
            algorithm = "SunX509";
        }

        SSLContext serverContext;
        SSLContext clientContext;
        try {
            KeyStore ks = KeyStore.getInstance("JKS");
            ks.load(SecureChatKeyStore.asInputStream(),
                    SecureChatKeyStore.getKeyStorePassword());

            // Set up key manager factory to use our key store
            KeyManagerFactory kmf = KeyManagerFactory.getInstance(algorithm);
            kmf.init(ks, SecureChatKeyStore.getCertificatePassword());

            // Initialize the SSLContext to work with our key managers.
            serverContext = SSLContext.getInstance(PROTOCOL);
            serverContext.init(kmf.getKeyManagers(), null, null);
        } catch (Exception e) {
            throw new Error(
                    "Failed to initialize the server-side SSLContext", e);
        }

        try {
            clientContext = SSLContext.getInstance(PROTOCOL);
            clientContext.init(null, SecureChatTrustManagerFactory.getTrustManagers(), null);
        } catch (Exception e) {
            throw new Error(
                    "Failed to initialize the client-side SSLContext", e);
        }

        SERVER_CONTEXT = serverContext;
        CLIENT_CONTEXT = clientContext;
    }

    public static SSLContext getServerContext() {
        return SERVER_CONTEXT;
    }

    public static SSLContext getClientContext() {
        return CLIENT_CONTEXT;
    }

    private SecureChatSslContextFactory() {
        // Unused
    }
}
```

12.4 Summary

这一章没有详细的按照netty in action书中来翻译，因为我感觉书中讲的很多都不是netty的重点，鄙人英文能实在有限，所以也就把精力不放在非核心上面了。若有读者需要详细在netty中使用spdy可以查看其它相关资料或文章，或者看本篇博文的例子代码。后面几章也会如此。