链滴

【开源推荐】SSLContext Kickstart 一个 SSL 工厂类库

作者：Jireh

原文链接：https://ld246.com/article/1629270893686

来源网站：链滴

SSLContext Kickstart 是一个库，它提供了一个高级 SSLFactory 类，用于配置 http 客户端以通过 SL/TLS 进行通信，以进行单向身份验证或双向身份验证。通过最小化外部依赖性，它被设计为尽可能量级。核心库仅依赖于 SLF4J 日志 API。

我是用在Netty支持wss协议，需要从本地路径读取pem格式的cert和key，生成SSLContext在netty使用。网上找了下netty的支持使用都是比较传统的用法，也没有发现pem格式的SSLContext生成用，所以特意找到了这个类库，发现还不错，才记录并分享下。

Github地址：https://github.com/Hakky54/sslcontext-kickstart

相关用法

基础

```
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.json.JSONException;
import org.json.JSONObject;

import nl.altindag.ssl.SSLFactory;

public class App {

    public static void main(String[] args) throws IOException, JSONException {
        SSLFactory sslFactory = SSLFactory.builder()
                .withDefaultTrustMaterial()
                .build();

        HttpClient httpClient = HttpClients.custom()
                .setSSLContext(sslFactory.getSslContext())
                .setSSLHostnameVerifier(sslFactory.getHostnameVerifier())
                .build();

        HttpGet request = new HttpGet("https://api.chucknorris.io/jokes/random");

        HttpResponse response = httpClient.execute(request);
        String chuckNorrisJoke = new JSONObject(EntityUtils.toString(response.getEntity())).getS
tring("value");

        System.out.println(String.format("Received the following status code: %d", response.getS
atusLine().getStatusCode()));
        System.out.println(String.format("Received the following joke: %s", chuckNorrisJoke));
    }

}
```

PEM

从类路径加载pem文件

```
X509ExtendedKeyManager keyManager = PemUtils.loadIdentityMaterial("certificate.pem", "pr
vate-key.pem");
X509ExtendedTrustManager trustManager = PemUtils.loadTrustMaterial("some-trusted-certif
cate.pem");

SSLFactory.builder()
        .withIdentityMaterial(keyManager)
        .withTrustMaterial(trustManager)
        .build();
```

netty wss demo

```
@Override
    protected void initChannel(NioSocketChannel ch) {
        ChannelPipeline pipeline = ch.pipeline();

        if (isWss) {
            X509ExtendedKeyManager keyManager = PemUtils.loadIdentityMaterial(Paths.get("/
ww/server/panel/vhost/cert/pioneer.ouhaihr.com/fullchain.pem"),
                    Paths.get("/www/server/panel/vhost/cert/pioneer.ouhaihr.com/privkey.pem"));

            SSLFactory sslFactory = SSLFactory.builder()
                .withIdentityMaterial(keyManager)
                .build();

            SSLEngine engine = sslFactory.getSslContext().createSSLEngine();
            engine.setUseClientMode(false);
            engine.setNeedClientAuth(false);
            pipeline.addLast(new SslHandler(engine));
        }

        // 处理第一次连接http的握手请求
        pipeline.addLast(new HttpServerCodec());
        // 写文件内容
        pipeline.addLast(new ChunkedWriteHandler());
        // 保证接收的http请求的完整性
        pipeline.addLast(new HttpObjectAggregator(maxContentLength));
        // 处理其他的WebSocketFrame
        pipeline.addLast(new WebSocketServerProtocolHandler(websocketPath));
        // 空闲检测
        ch.pipeline().addLast(new MyIdleStateHandler());
        // WebSocket数据包编解码器
        ch.pipeline().addLast(webSocketPacketCodec);
        // 添加tcp/websocket通用handler
        pipelineUtil.addHandler(pipeline);
    }
```

更多的使用示例请到GIthub查看