Changeset 124 for projects/mobile/meego/whatsapp/symbian-s40/v2.0.7/src/com/whatsapp/client/FunXMPP.java

Timestamp: 02/20/12 22:25:17 (less than one hour ago)
Author: user

Message:
File: 🟨 1 edited

🟨 projects/mobile/meego/whatsapp/symbian-s40/v2.0.7/src/com/whatsapp/client/FunXMPP.java (33 diffs)

☐ Unmodified 🟩 Added 🟥 Removed

projects/mobile/meego/whatsapp/symbian-s40/v2.0.7/src/com/whatsapp/client/FunXMPP.java

Tabular | Unified Tabular | Unified

```
r123 r124
   3    3 /*      */ import com.whatsapp.api.util.MessageDigest;
   4    4 /*      */ import com.whatsapp.api.util.Utilities;
        5 import com.whatsapp.api.util.S40MD5Digest;
   5    6 /*      */ import com.whatsapp.org.bouncycastle.util.encoders.Base64;
   6    7 /*      */ import com.whatsapp.org.it.yup.xml.KXmlParser;
 ...  ...
  22   23 /*      */ import java.util.Random;
  23   24 /*      */ import java.util.Vector;
       25            import java.net.*;
  24   26 /*      */
  25   27 /*      */ public class FunXMPP
  26   28 /*      */ {
       29
       30                 public static void main (String []args) throws Exception
       31                 {
       32                         System.out.println("HELLO FUNXMPP!");
       33                          String usePushName = "s40 user";
       34                         String resource = "S40-1.2-443";
       35                         String chatUserID="PHONE_NUMBER_HERE";
       36                         String domain ="s.whatsapp.net";
       37                         String socketURL="bin-short.whatsapp.net:5222";
       38                          FunXMPP.Connection connection = null;
       39
       40                 /* 105 */     Socket conn = null;
       41                 /* 106 */     InputStream is = null;
       42                 /* 107 */     OutputStream os = null;
       43
       44
       45                          conn = new Socket("bin-short.whatsapp.net",5222);
       46
       47                           is = conn.getInputStream();
       48 /* 177 */                os = conn.getOutputStream();
       49                  Object login = new FunXMPP.Login.WhatsApp(new FunXMPP.BinTreeNodeReader(is, FunXMPP.dictionary), new
          FunXMPP.BinTreeNodeWriter(os, FunXMPP.dictionary), new S40MD5Digest());
       50 /*     */
       51 /* 191 */         connection = new FunXMPP.Connection((FunXMPP.Login)login, "s.whatsapp.net", resource, chatUserID,
          usePushName, "8ef85f123674647e9b421a8e4bea63cb", null,null,null);
       52 /*     */
       53 /* 196 */         ((FunXMPP.Login)login).setConnection(connection);
       54
       55                          connection.setReceiptAckCapable(true);
       56                          connection.login();
       57                 }
  27   58 /*   28 */   public static FunStore message_store = new FunStore();
  28   59 /*      */
 ...  ...
 221  252 /*      */        }
 222  253 /* 3071 */        return nextInternalTree(); } catch (XmlPullParserException e) {
 223      /*      */      }
 224      /* 3073 */      throw new FunXMPP.CorruptStreamException("got XmlPullParseX : " + e.toString());
      254                                throw new FunXMPP.CorruptStreamException("got XmlPullParseX : " + e.toString());
      255 /*      */      }
      256 /* 3073 */
 225  257 /*      */    }
 226  258 /*      */
 ...  ...
 354  386 /*      */    void sendTextConvertEntities(String s) throws IOException {
 355  387 /* 3018 */      int p1 = 0;
 356      /*      */
 357      /* 3020 */      for (int p2 = 0; p2 < s.length(); p2++) {
      388 /*      */      int p2 = 0;
      389 /* 3020 */      for (p2 = 0; p2 < s.length(); p2++) {
 358  390 /* 3021 */        String entity = getEntity(s.charAt(p2));
 359  391 /* 3022 */        if (entity != null) {
 ...  ...
 387  419 /*      */    InputStream rawIn;
 388  420 /*      */    InputStream in;
 389      /* 2748 */    byte[] buf = new byte['Ã¿'];
      421 /* 2748 */    //byte[] buf = new byte['Ã¿'];
      422             byte[] buf = new byte[1024];
 390  423 /* 2749 */    int bufSize = 0;
 391  424 /*      */     String[] tokenMap;
 ...  ...
 475  508 /*      */       else
 476  509 /*      */       {
 477      /*      */         int size;
      510 /*      */         size = 0;
 478  511 /* 2830 */         if (token == 248) {
 479  512 /* 2831 */           size = readInt8(this.in);
 ...  ...
 481  514 /*      */         else
 482  515 /*      */         {
 483      /*      */           int size;
      516 /*      */           size = 0;
 484  517 /* 2832 */           if (token == 249)
 485  518 /* 2833 */             size = readInt16(this.in);
 ...  ...
 488  521 /*      */         }
 489  522 /*      */       }
 490      /*      */      int size;
      523 /*      */
 491  524 /* 2837 */       return size;
 492  525 /*      */     }
 ...  ...
 544  577 /* 2890 */       while (count < len)
 545  578 /* 2891 */         count += i.read(buf, count, len - count);
      579
      580
 546  581 /*      */     }
 547  582 /*      */
 ...  ...
 550  585 /*      */     {
 551  586 /*      */       String ret;
 552      /*      */       String ret;
      587 /*      */
 553  588 /* 2897 */       if ((token >= 0) && (token < this.tokenMap.length))
 554  589 /* 2898 */         ret = this.tokenMap[token];
 ...  ...
 569  604 /* 2913 */       int intTop = i.read();
 570  605 /* 2914 */       int intBot = i.read();
      606
 571  607 /* 2915 */       int value = (intTop << 8) + intBot;
 572  608 /* 2916 */       return value;
 ...  ...
 591  628 /*      */       }
 592  629 /* 2936 */       int attribCount = (size - 2 + size % 2) / 2;
      630
 593  631 /* 2937 */       FunXMPP.KeyValue[] attribs = readAttributes(attribCount);
 594  632 /*      */     }
 595  633 /*      */
 596  634 /*      */     private void fillBuffer(int stanzaSize) throws IOException {
      635
      636
 597  637 /* 2941 */       if (this.buf.length < stanzaSize) {
 598  638 /* 2942 */         int newsize = Math.max(this.buf.length * 3 / 2, stanzaSize);
 ...  ...
 602  642 /* 2946 */       fillArray(this.buf, stanzaSize, this.rawIn);
 603  643 /* 2947 */       this.in = new ByteArrayInputStream(this.buf, 0, stanzaSize);
 604      /*      */     }
      644
      645
      646
      647 /*      */     }
      648
 605  649 /*      */   }
 606  650 /*      */
 ...  ...
 896  940 /* 2104 */       this.digest = d;
 897  941 /*      */     }
      942
      943  public Login(FunXMPP.TreeNodeReader r, FunXMPP.TreeNodeWriter w)
      944 /*      */     {
      945 /* 2102 */       this.in = r;
      946 /* 2103 */       this.out = w;
      947              this.digest = new S40MD5Digest();
      948 /* 2104 */
      949 /*      */     }
 898  950 /*      */     public void setConnection(FunXMPP.Connection c) {
 899  951 /* 2107 */       this.connection = c;
 ...  ...
 923  975 /*      */
 924  976 /* 2133 */       ByteArrayOutputStream bos = new ByteArrayOutputStream();
 925      /*      */
      977 /*      */
      978              System.out.println(md5Digest((this.connection.user + ":" + this.connection.domain + ":" +
this.connection.password).getBytes()));
 926  979 /* 2135 */       bos.write(md5Digest((this.connection.user + ":" + this.connection.domain + ":" +
this.connection.password).getBytes()));
 927  980 /* 2136 */       bos.write(58);
```

```
/* 2158 */       bos.write(58);
...
/*      */
/* 2159 */       String KD = new String(bytesToHex(md5Digest(A1))) + ":" + nonce + ":" + nc + ":" + cnonce + ":auth:" + new String(bytesToHex(md5Digest(A2.getBytes()))));

                //System.out.println(KD);System.exit(1);
/* 2160 */       String response = new String(bytesToHex(md5Digest(KD.getBytes())));
/*      */
...
/* 2176 */       bigger_response.append("\",nc=");
/* 2177 */       bigger_response.append(nc);

                //System.out.println(bigger_response.toString());System.exit(1);
/* 2178 */       return bigger_response.toString(); }
/*      */
...
/*      */
/* 2396 */         this.in.streamStart();

/* 2397 */         System.err.println("read stream start");
/* 2398 */         String challengeData = readFeaturesAndChallenge();
/* 2399 */         System.err.println("read features and challenge");
/*      */
/*      */         //System.out.println(challengeData);
                   //System.exit(1);
/* 2401 */         sendResponse(challengeData);


/* 2402 */         System.err.println("sent response");
/* 2403 */         readSuccess();
...
/* 2424 */         while ((root = this.in.nextTree()) != null) {
/* 2425 */           if (FunXMPP.ProtocolTreeNode.tagEquals(root, "stream:features")) {

/* 2426 */             server_supports_receipt_acks = root.getChild("receipt_acks") != null; continue;
/* 2427 */           }if (FunXMPP.ProtocolTreeNode.tagEquals(root, "challenge")) {

/* 2428 */             this.connection.supports_receipt_acks = ((this.connection.supports_receipt_acks) && (server_supports_receipt_acks));
/* 2429 */             String data = new String(Base64.decode(root.data.getBytes()));

/* 2430 */             return data;
/*      */           }
...
/*      */       private void readSuccess() throws FunXMPP.CorruptStreamException, IOException, FunXMPP.LoginFailureException {
/* 2444 */         FunXMPP.ProtocolTreeNode node = this.in.nextTree();

                   System.out.println(node.tag);
                  // System.exit(1);
/* 2445 */         if (FunXMPP.ProtocolTreeNode.tagEquals(node, "failure")) {
/* 2446 */           throw new FunXMPP.LoginFailureException(0);
...
/*  785 */       FunXMPP.KeyValue type = new FunXMPP.KeyValue("type", FunXMPP.FMessage.getMessage_WA_Type_StrValue(message.media_wa_type));
/*      */       FunXMPP.KeyValue[] mediaAttributes;
/*      */       FunXMPP.KeyValue[] mediaAttributes;
/*      */
/*  787 */       if (5 == message.media_wa_type) {
/*  788 */         mediaAttributes = new FunXMPP.KeyValue[] { xmlns, type, new FunXMPP.KeyValue("latitude", Double.toString(message.latitude)), new FunXMPP.KeyValue("longitude", Double.toString(message.longitude)) };
...
/*      */       else
/*      */       {
/*      */         FunXMPP.KeyValue[] mediaAttributes;
/*      */
/*  791 */         if ((4 != message.media_wa_type) && (message.media_name != null) && (message.media_url != null) && (message.media_size > 0L))
/*      */         {
/*      */           FunXMPP.KeyValue[] mediaAttributes;
/*      */
/*  795 */           if (message.media_duration_seconds <= 0) {
/*  796 */             mediaAttributes = new FunXMPP.KeyValue[] { xmlns, type, new FunXMPP.KeyValue("file", message.media_name), new FunXMPP.KeyValue("size", Long.toString(message.media_size)), new FunXMPP.KeyValue("url", message.media_url) };
...
/*      */       }
/*      */       FunXMPP.ProtocolTreeNode mediaNode;
/*      */       FunXMPP.ProtocolTreeNode mediaNode;
/*      */
/*  811 */       if ((4 == message.media_wa_type) && (message.media_name != null))
/*      */       {
...
/*  967 */       this.iqid += 1;
/*      */       String id;
/*      */       String id;
/*      */
/*  969 */       if (this.verbose_id)
/*  970 */         id = prefix + this.iqid;
...
/* 1185 */       String id = makeId("privacy_");
/*      */ 
/* 1187 */       this.pending_server_requests.put(id, new FunXMPP.IqResultHandler(onSuccess, onError) { private final Runnable val$onSuccess;
/*      */         private final FunXMPP.IntRunnable val$onError;
/*      */ 
/* 1190 */         public void parse(FunXMPP.ProtocolTreeNode node, String from) throws IOException, FunXMPP.CorruptStreamException { if (this.val$onSuccess != null)
/* 1191 */             this.val$onSuccess.run();
/*      */         }
/*      */ 
/*      */         public void error(int code)
/*      */         {
/* 1196 */           if (this.val$onError != null)
/* 1197 */             this.val$onError.run(code);
/*      */         }
/*      */       });
/* 1201 */       FunXMPP.ProtocolTreeNode[] listNodeChildren = new FunXMPP.ProtocolTreeNode[list.size()];
/* 1202 */       Enumeration e = list.elements();
...
/* 1314 */       String id = makeId("create_group_");
/*      */ 
/* 1316 */       this.pending_server_requests.put(id, new FunXMPP.IqResultHandler(subject, onSuccess, onError) { private final String val$subject;
/*      */         private final FunXMPP.StringRunnable val$onSuccess;
/*      */         private final FunXMPP.IntRunnable val$onError;
/*      */ 
/* 1318 */         public void parse(FunXMPP.ProtocolTreeNode node, String from) throws FunXMPP.CorruptStreamException, IOException { FunXMPP.ProtocolTreeNode groupNode = node.getChild(0);
/* 1319 */           FunXMPP.ProtocolTreeNode.require(groupNode, "group");
/* 1320 */           String gid = groupNode.getAttributeValue("id");
/* 1321 */           FunXMPP.Connection.this.group_event_handler.onGroupCreated(FunXMPP.Connection.this.gidToGjid(gid), this.val$subject);
/* 1322 */           if (this.val$onSuccess != null)
/* 1323 */             this.val$onSuccess.run(FunXMPP.Connection.this.gidToGjid(gid)); }
/*      */ 
/*      */         public void error(int code)
/*      */         {
/* 1327 */           if (this.val$onError != null)
/* 1328 */             this.val$onError.run(code);
/*      */         }
/*      */       });
/* 1316 */
/* 1332 */       FunXMPP.ProtocolTreeNode groupNode = new FunXMPP.ProtocolTreeNode("group", new FunXMPP.KeyValue[] { new FunXMPP.KeyValue("xmlns", "w:g"), new FunXMPP.KeyValue("action", "create"), new FunXMPP.KeyValue("subject", subject) });
/*      */ 
...
/* 1352 */       String id = makeId("remove_group_");
/*      */ 
/* 1354 */       this.pending_server_requests.put(id, new FunXMPP.IqResultHandler(onSuccess, onError) { private final Runnable val$onSuccess;
/*      */         private final FunXMPP.IntRunnable val$onError;
/*      */ 
/* 1356 */         public void parse(FunXMPP.ProtocolTreeNode node, String from) throws FunXMPP.CorruptStreamException, IOException { if (this.val$onSuccess != null)
/* 1357 */             this.val$onSuccess.run(); }
/*      */ 
/*      */         public void error(int code)
/*      */         {
/* 1361 */           if (this.val$onError != null)
/* 1362 */             this.val$onError.run(code);
/*      */         }
/*      */       });

/* 1366 */       FunXMPP.ProtocolTreeNode groupNode = new FunXMPP.ProtocolTreeNode("group", new FunXMPP.KeyValue[] { new FunXMPP.KeyValue("xmlns", "w:g"), new FunXMPP.KeyValue("action", "delete") });
/*      */ 
...
/* 1427 */       String id = makeId("get_groups_");
/*      */ 
/* 1429 */       this.pending_server_requests.put(id, new FunXMPP.IqResultHandler(onSuccess, onError) { private final Runnable val$onSuccess;
/*      */         private final FunXMPP.IntRunnable val$onError;
/*      */ 
/* 1431 */         public void parse(FunXMPP.ProtocolTreeNode node, String from) throws FunXMPP.CorruptStreamException, IOException { Vector groups = new Vector();
/* 1432 */           FunXMPP.Connection.this.readGroupList(node, groups);
/* 1433 */           FunXMPP.Connection.this.group_event_handler.onParticipatingGroups(groups);
/* 1434 */           if (this.val$onSuccess != null)
/* 1435 */             this.val$onSuccess.run();
/*      */         }
/*      */ 
/*      */         public void error(int code)
/*      */         {
/* 1440 */           if (this.val$onError != null)
/* 1441 */             this.val$onError.run(code);
/*      */         }
/*      */       });
/* 1429 */
/* 1446 */       sendGetGroups(id, "participating");
/*      */     }
...
/* 1469 */       String id = makeId("set_group_subject_");
/*      */
/* 1471 */       this.pending_server_requests.put(id, new FunXMPP.IqResultHandler(onSuccess, onError) { private final Runnable val$onSuccess;
/*      */         private final FunXMPP.IntRunnable val$onError;
/*      */
/* 1473 */         public void parse(FunXMPP.ProtocolTreeNode node, String from) throws FunXMPP.CorruptStreamException, IOException { FunXMPP.Connection.this.group_event_handler.onSetSubject(from);
/* 1474 */           if (this.val$onSuccess != null)
/* 1475 */             this.val$onSuccess.run(); }
/*      */
/*      */         public void error(int code)
/*      */         {
/* 1479 */           if (this.val$onError != null)
/* 1480 */             this.val$onError.run(code);
/*      */         }
/*      */       });
/* 1471 */
/* 1484 */       FunXMPP.ProtocolTreeNode subjectNode = new FunXMPP.ProtocolTreeNode("subject", new FunXMPP.KeyValue[] { new FunXMPP.KeyValue("xmlns", "w:g"), new FunXMPP.KeyValue("value", subject) });
/*      */
...
/* 1502 */       String id = makeId("add_group_participants_");
/*      */
/* 1504 */       this.pending_server_requests.put(id, new FunXMPP.IqResultHandler(onSuccess, onError) { private final Runnable val$onSuccess;
/*      */         private final FunXMPP.IntRunnable val$onError;
/*      */
/* 1506 */         public void parse(FunXMPP.ProtocolTreeNode node, String from) throws FunXMPP.CorruptStreamException, IOException { Vector successVector = new Vector();
/* 1507 */           Hashtable failTable = new Hashtable();
/* 1508 */           FunXMPP.Connection.this.readSuccessAndFailure(node, successVector, failTable, "add");
/* 1509 */           FunXMPP.Connection.this.group_event_handler.onAddGroupParticipants(from, successVector, failTable);
/* 1510 */           if (this.val$onSuccess != null)
/* 1511 */             this.val$onSuccess.run(); }
/*      */
/*      */         public void error(int code)
/*      */         {
/* 1515 */           if (this.val$onError != null)
/* 1516 */             this.val$onError.run(code);
/*      */         }
/*      */       });
/* 1504 */
/* 1521 */       sendVerbParticipants(gjid, participants, id, "add");
/*      */     }
...
/* 1528 */       String id = makeId("remove_group_participants_");
/*      */
/* 1530 */       this.pending_server_requests.put(id, new FunXMPP.IqResultHandler(onSuccess, onError) { private final Runnable val$onSuccess;
/*      */         private final FunXMPP.IntRunnable val$onError;
/*      */
/* 1532 */         public void parse(FunXMPP.ProtocolTreeNode node, String from) throws FunXMPP.CorruptStreamException, IOException { Vector successVector = new Vector();
/* 1533 */           Hashtable failTable = new Hashtable();
/* 1534 */           FunXMPP.Connection.this.readSuccessAndFailure(node, successVector, failTable, "remove");
/* 1535 */           FunXMPP.Connection.this.group_event_handler.onRemoveGroupParticipants(from, successVector, failTable);
/* 1536 */           if (this.val$onSuccess != null)
/* 1537 */             this.val$onSuccess.run(); }
/*      */
/*      */         public void error(int code)
/*      */         {
/* 1541 */           if (this.val$onError != null)
/* 1542 */             this.val$onError.run(code);
/*      */         }
/*      */       });
/* 1530 */
/* 1547 */       sendVerbParticipants(gjid, participants, id, "remove");
/*      */     }
...
/* 1561 */       String id = makeId("leave_group_");
/*      */
/* 1563 */       this.pending_server_requests.put(id, new FunXMPP.IqResultHandler(onSuccess, onError) { private final Runnable val$onSuccess;
/*      */         private final FunXMPP.IntRunnable val$onError;
/*      */
/* 1565 */         public void parse(FunXMPP.ProtocolTreeNode node, String from) { FunXMPP.ProtocolTreeNode leaveChild = node.getChild("leave");
/* 1566 */           if (leaveChild != null) {
/* 1567 */             Vector groups = leaveChild.getAllChildren("group");
/* 1568 */             for (int i = 0; i < groups.size(); i++) {
/* 1569 */ FunXMPP.Connection.this.group_event_handler.onLeaveGroup(((FunXMPP.ProtocolTreeNode)groups.elementAt(i)).getAttributeValue("id"));
/*      */             }
/*      */           }
/*      */
/* 1573 */           if (this.val$onSuccess != null)
/* 1574 */             this.val$onSuccess.run(); }
/*      */
/*      */         public void error(int code)
/*      */         {
/* 1578 */           if (this.val$onError != null)
/* 1579 */             this.val$onError.run(code);
/*      */         }
/*      */       });
/* 1563 */
/* 1583 */       int size = gjids.size();
/* 1584 */       FunXMPP.ProtocolTreeNode[] groupJidList = new FunXMPP.ProtocolTreeNode[size];
...
/*      */     public abstract void parse(FunXMPP.ProtocolTreeNode paramProtocolTreeNode, String paramString)
/*      */       throws IOException, FunXMPP.CorruptStreamException;

                public  IqResultHandler()
                {
                }
                public  IqResultHandler(java.lang.Runnable x,com.whatsapp.client.FunXMPP.IntRunnable y)
                {

                }

                public  IqResultHandler(java.lang.String x,com.whatsapp.client.FunXMPP.StringRunnable
y,com.whatsapp.client.FunXMPP.IntRunnable z)
                {

                }
/*      */
/*      */     public void error(int code)
```