Tipps & Tricks: UTL_SMTP

Bereich:	PL/SQL	Erstellung:	02/2002 HA
Versionsinfo:	8.1, 9.2, 10.1, 10.2, 11.1, 11.2	Letzte Überarbeitung:	06/2009 MP

UTL_SMTP

Ab Version 8i ist es möglich, mit dem Package UTL_SMTP aus PL/SQL heraus E-Mails zu versenden, falls ein SMTP-Server zur Verfügung steht. Da das Original-SMTP-Protokoll einen 7-Bit ASCII-Zeichensatz verwendet, wird jeder Text vor dem Versenden in US7ASCII umgewandelt.

Fast alle Routinen sind sowohl als Prozeduren als auch als Funktionen verfügbar. Die Funktionen haben normalerweise als Returnwert einen Record vom Typ UTL_SMTP.reply, der den Returncode des SMTP-Servers beinhaltet. Arbeitet man mit den Funktionen, so kann man diesen überprüfen und bei fehlerhaftem Returnwert eine entsprechende Fehlermeldung ausgeben. Die Prozeduren dagegen liefern keinen Returncode zurück; stattdessen wird durch die Prozeduren eine Exception ausgelöst, falls ein Fehlercode auftrat.

UTL_SMTP.connection

Ein Record dieses Typs repräsentiert die Verbindung zum SMTP-Server. Sie ist in jeden Funktions- bzw. Prozeduraufruf als Parameter enthalten. Als Parameter <host> ist dabei entweder der volle Domänenname oder die IP-Adresse möglich.

Nötige Prozedur- bzw. Funktionsaufrufe:

- Bei der Versendung ist folgende Reihenfolge einzuhalten:
- Aufbau der Verbindung: UTL_SMTP.open_connection
- Handshaking: UTL_SMTP.helo
- Absender bekanntmachen: UTL_SMTP.mail
- Empfänger bekanntmachen: UTL_SMTP.rcpt
- Mail versenden: UTL_SMTP.data
- Verbindung beenden: UTL_SMTP.quit

Sollen mehrere Zeilen versendet werden, so muss am Ende jeder Zeile ein CRLF (CHR(13)||CHR(10)) stehen. Statt eines einmaligen Aufrufs von UTL_SMTP.data, an das nur eine einzige Variable vom Typ VARCHAR2 als Text der E-Mails übergeben werden kann, können auch folgende Programmeinheiten aufgerufen werden:

- Beginn des Mail-Body: UTL_SMTP.open_data
- Mail-Body schreiben: UTL_SMTP.write_data
- Beendigung des Mail-Body: UTL_SMTP.close_data

UTL_SMTP.write_data kann mehrfach aufgerufen werden und hängt immer am Ende etwas an, beinhaltet aber auch kein CRLF. Da der Server erst am Ende einen Returncode liefert, sind open_data und write_data nur als Prozeduren implementiert.

Der Empfänger muss zwar mit rcpt bekanntgemacht werden, aber trotzdem kommt die E-Mail eventuell nicht an, wenn nicht im Body zusätzlich "To: empfänger<empfänger>" aufgeführt wird. Soll auch die Betreff-Zeile der Mail gefüllt werden, so ist dies mit "Subject: betreff" möglich.

Version 11g: Ab Version 11.1 muss der Netzwerkzugriff auf den SMTP-Server über eine Access Control List freigegeben werden.

Beispiel:

```
PROCEDURE SMTP_HOME
   IS
    v_sender     VARCHAR2(2000) := 'sender@muniqsoft.de';
    v_rec        VARCHAR2(2000) := 'receiver@muniqsoft.de';
    v_host       VARCHAR2(2000) := 'mailserver.muniqsoft.de';
    v_con        UTL_SMTP.connection;
    v_ret        UTL_SMTP.reply;
   v_smtp_pwd   VARCHAR2(2000) := '???';
    v_message    VARCHAR2(2000);
    crlf         VARCHAR2(2) := chr(13)||chr(10);
BEGIN

    v_ret := UTL_SMTP.open_connection(v_host, 25, v_con);
    if v_ret.code <>220 then
        DBMS_OUTPUT.put_line
         ('Verbindung konnte nicht hergestellt werden');
        UTL_SMTP.quit(v_con);
        RETURN;
    end if;

   -- Die folgenden 3 Zeilen müssen nur bei einer
   -- Authentifizierung des SMTP Servers verwendet werden
    v_smtp_pwd:=utl_raw.cast_to_varchar2(v_smtp_pwd);
    utl_smtp.command( v_con, 'AUTH LOGIN');
    utl_smtp.command( v_con, utl_raw.cast_to_varchar2(
            utl_encode.base64_encode( utl_raw.cast_to_raw( v_sender ))) );
    utl_smtp.command( v_con, utl_raw.cast_to_varchar2(
            utl_encode.base64_encode( utl_raw.cast_to_raw(v_smtp_pwd))) );

    v_ret := UTL_SMTP.helo(v_con, v_host);
    if v_ret.code <>250 then
        DBMS_OUTPUT.put_line
             ('Handshaking konnte nicht durchgeführt werden');
        UTL_SMTP.quit(v_con);
        RETURN;
    end if;

    v_ret := UTL_SMTP.mail(v_con, v_sender);
    if v_ret.code <>250 then
        DBMS_OUTPUT.put_line('Initialisierung ergab Fehler');
        UTL_SMTP.quit(v_con);
        RETURN;
    end if;

    v_ret := UTL_SMTP.rcpt(v_con, v_rec);
    if v_ret.code <>250 and v_ret.code <> 251 then
        DBMS_OUTPUT.put_line
             ('Recipient-Initialisierung ergab Fehler');
        UTL_SMTP.quit(v_con);
```

```
        RETURN;
    end if;

    v_message := v_message||'To: '||v_rec||'<'||v_rec||'>'||crlf;
    v_message := v_message||'Subject: UTL_SMTP'||crlf;
    v_message := v_message||'Hier steht der Text';
    v_ret := UTL_SMTP.data(v_con, v_message);

    if v_ret.code <>250 then
        DBMS_OUTPUT.put_line('Versendung fehlgeschlagen');
        UTL_SMTP.quit(v_con);
        RETURN;
    end if;

    v_ret := UTL_SMTP.quit(v_con);
EXCEPTION

    WHEN others THEN
        UTL_SMTP.quit(v_con);
        DBMS_OUTPUT.put_line(sqlerrm) ;
END; -- Procedure SMTP_HOME
```

Beispiel prozedural und mit write_data:

```
Procedure SMTP_HOME_P
   IS
    v_sender VARCHAR2(2000) := 'sender@muniqsoft.de';
    v_rec    VARCHAR2(2000) := 'sender@muniqsoft.de';
    v_host   VARCHAR2(2000) := 'mailserver.muniqsoft.de';
    v_con    UTL_SMTP.connection;
    crlf     VARCHAR2(2) := chr(13)||chr(10);

BEGIN
    v_con := UTL_SMTP.open_connection(v_host, 25);
    UTL_SMTP.helo(v_con, v_host);
    UTL_SMTP.mail(v_con, v_sender);
    UTL_SMTP.rcpt(v_con, v_rec);
    UTL_SMTP.open_data(v_con);
    UTL_SMTP.write_data(v_con,'To: '||v_rec||'<'||v_rec||'>'||crlf);
    UTL_SMTP.write_data(v_con,'Subject: UTL_SMTP'||crlf);
    UTL_SMTP.write_data(v_con,'Hier steht der Text');
    UTL_SMTP.close_data(v_con);
    UTL_SMTP.quit(v_con);

EXCEPTION
    WHEN others THEN
        UTL_SMTP.quit(v_con);
        DBMS_OUTPUT.put_line(sqlerrm) ;
END; -- Procedure SMTP_HOME_P
```