```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Text.RegularExpressions;

namespace MessageLogger
{
    /// <summary>
    /// This class handles all communication with the SQL server. For security
    /// we will only allow stored procedures to be executed via the web application.
    ///
    /// If you want to extend the functionality of the web application, create new
    /// stored procedures and create a new associated function in this class.
    /// </summary>
    public class spExecution
    {
        private System.Data.SqlClient.SqlConnection sqlConn;
        // Regular expression to verify string is alphanumeric and no longer than 50 chars.
        Regex reg = new Regex(@"^[a-zA-Z0-9'.\s]{1,50}$");

        /// <summary>
        /// Default constructor. This does nothing.
        /// </summary>
        public spExecution()
        {
        }

        /// <summary>
        /// Connects to the default localhost database.
        /// </summary>
        public void SQLConnect()
        {
            sqlConn = new System.Data.SqlClient.SqlConnection();
            sqlConn.ConnectionString = "server=.; database=MLogger; uid=mlogger; pwd=trustera↙
    ;";
            sqlConn.Open();
        }

        /// <summary>
        /// Overloaded SQL connection function that allows one to specify a specific server,
        /// database, user, and password.
        /// </summary>
        /// <param name="server">The server IP address or NetBIOS name.</param>
        /// <param name="database">The database we want to use</param>
        /// <param name="user">The database user</param>
        /// <param name="password">The database user's password</param>
        public void SQLConnect(String server, String database, String user, String password)
        {
            sqlConn = new System.Data.SqlClient.SqlConnection();
            sqlConn.ConnectionString = "server=" + server + "; database=" + database + "; uid↙
    =" + user + "; pwd=" + password + ";";
            sqlConn.Open();
        }

        public void SQLDisconnect()
        {
            sqlConn.Close();
        }

        /// <summary>
        /// Validates a user against the Users table.
        /// </summary>
        /// <param name="userName">User's login name</param>
        /// <param name="password">User's login password</param>
        /// <returns>1 if valid, 0 if invalid</returns>
        public String ValidateUser(String userName, String password)
        {
            userName = CleanSQL(userName);
            password = CleanSQL(password);

            // The stored procedure we're executing
            System.Data.SqlClient.SqlCommand sqlCmd = new System.Data.SqlClient.SqlCommand(   ↙
    "spValidateUser", sqlConn);
            sqlCmd.CommandType = System.Data.CommandType.StoredProcedure;

            // Add our parameters to the stored procedure
            sqlCmd.Parameters.Add("@userName", SqlDbType.VarChar).Value = userName;
            sqlCmd.Parameters.Add("@password", SqlDbType.VarChar).Value = password;

            // The return value for the stored procedure
            System.Data.SqlClient.SqlParameter pRetValue = sqlCmd.Parameters.Add("@Ret",      ↙
    SqlDbType.Int);
            pRetValue.Direction = ParameterDirection.ReturnValue;

            // Execute the stored procedure
            sqlCmd.ExecuteScalar();

            return sqlCmd.Parameters["@Ret"].Value.ToString();
        }

        /// <summary>
        /// Creates a new session ID for the user
        /// </summary>
        /// <param name="userName">The user for which the session is being opened</param>
        /// <returns>The new session ID</returns>
        public long NewSessionID(String userName)
        {
            long sessionID;
            userName = CleanSQL(userName);

            ClearExpiredSessions(); // clear all expired sessions

            // The stored procecure we're executing
            System.Data.SqlClient.SqlCommand sqlCmd = new System.Data.SqlClient.SqlCommand(   ↙
    "spNewSessionID", sqlConn);
            sqlCmd.CommandType = System.Data.CommandType.StoredProcedure;

            // Add our parameters to the stored procedure
            sqlCmd.Parameters.Add("@userName", SqlDbType.VarChar).Value = userName;

            // The return value for the stored procedure
            System.Data.SqlClient.SqlParameter pRetValue = sqlCmd.Parameters.Add("@Ret",      ↙
    SqlDbType.Int);
            pRetValue.Direction = ParameterDirection.ReturnValue;

            // Execute the stored procedure
            sqlCmd.ExecuteScalar();

            // Convert to long
            long.TryParse(sqlCmd.Parameters["@Ret"].Value.ToString(), out sessionID);

            return sessionID;
        }

        /// <summary>
        /// Checks whether the session ID is valid
        /// </summary>
        /// <param name="sessionID">The user's session ID</param>
        /// <returns>TRUE if valid, FALSE if invalid</returns>
        public bool ValidateSessionID(long sessionID)
        {
            ClearExpiredSessions(); // clear all expired sessions

            // The stored procecure we're executing
            System.Data.SqlClient.SqlCommand sqlCmd = new System.Data.SqlClient.SqlCommand( ↙
    "spValidateSession", sqlConn);
            sqlCmd.CommandType = System.Data.CommandType.StoredProcedure;

            // Add our parameters to the stored procedure
            sqlCmd.Parameters.Add("@sessionID", SqlDbType.Int).Value = sessionID;

            // The return value for the stored procedure
            System.Data.SqlClient.SqlParameter pRetValue = sqlCmd.Parameters.Add("@Ret",     ↙
    SqlDbType.Int);
            pRetValue.Direction = ParameterDirection.ReturnValue;

            // Execute the stored procedure
            sqlCmd.ExecuteScalar();

            // Verify that the session ID is valid
            if (sqlCmd.Parameters["@Ret"].Value.ToString() == "1")
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        /// <summary>
        /// Posts a new message into the database
        /// </summary>
        /// <param name="sessionID">The user's session ID</param>
        /// <param name="title">Title of the new message</param>
        /// <param name="message">Message text</param>
        /// <returns>messageID of the new message</returns>
        public int PostNewMessage(long sessionID, String title, String message)
        {
            int messageID; // the new message ID

            // The stored procecure we're executing
            System.Data.SqlClient.SqlCommand sqlCmd = new System.Data.SqlClient.SqlCommand( ↙
    "spPostNewMessage", sqlConn);
            sqlCmd.CommandType = System.Data.CommandType.StoredProcedure;

            // Add our parameters to the stored procedure
            sqlCmd.Parameters.Add("@sessionID", SqlDbType.Int).Value = sessionID;
            sqlCmd.Parameters.Add("@title", SqlDbType.VarChar).Value = title;
            sqlCmd.Parameters.Add("@message", SqlDbType.VarChar).Value = message;

            // The return value for the stored procedure
            System.Data.SqlClient.SqlParameter pRetValue = sqlCmd.Parameters.Add("@Ret",     ↙
    SqlDbType.Int);
            pRetValue.Direction = ParameterDirection.ReturnValue;

            // Execute the stored procedure
            sqlCmd.ExecuteScalar();

            // Convert the return string to int
            int.TryParse(sqlCmd.Parameters["@Ret"].Value.ToString(), out messageID);

            return messageID;
        }

        public void PostMsgAttachment(long sessionID, int messageID, String mediaType, String↙
     mediaLocation)
        {
            // The stored procecure we're executing
            System.Data.SqlClient.SqlCommand sqlCmd = new System.Data.SqlClient.SqlCommand(  ↙
    "spPostMsgAttachment", sqlConn);
            sqlCmd.CommandType = System.Data.CommandType.StoredProcedure;

            // Add our parameters to the stored procedure
            sqlCmd.Parameters.Add("@sessionID", SqlDbType.Int).Value = sessionID;
            sqlCmd.Parameters.Add("@messageID", SqlDbType.VarChar).Value = messageID;
            sqlCmd.Parameters.Add("@mediaType", SqlDbType.VarChar).Value = mediaType;
            sqlCmd.Parameters.Add("@mediaLocation", SqlDbType.VarChar).Value = mediaLocation;

            // Execute the stored procedure
            sqlCmd.ExecuteScalar();
        }

        /// <summary>
        /// Gets the user name associated with a session ID (this is used mostly for file    ↙
    storage)
        /// </summary>
        /// <param name="sessionID">The user's current session ID</param>
        /// <returns>The user's user name</returns>
        public String GetUserFromSession(long sessionID)
        {
            // The stored procecure we're executing
            System.Data.SqlClient.SqlCommand sqlCmd = new System.Data.SqlClient.SqlCommand(  ↙
    "spGetUserFromSession", sqlConn);
            sqlCmd.CommandType = System.Data.CommandType.StoredProcedure;

            // Add our parameters to the stored procedure
            sqlCmd.Parameters.Add("@sessionID", SqlDbType.Int).Value = sessionID;

            // Execute the stored procedure
            System.Data.SqlClient.SqlDataReader reader = sqlCmd.ExecuteReader();

            reader.Read();

            return reader["UserName"].ToString();
        }

        /// <summary>
        /// Clears all expired sessions from the database
        /// </summary>
        private void ClearExpiredSessions()
        {
            // The stored procedure we're executing
            System.Data.SqlClient.SqlCommand sqlCmd = new System.Data.SqlClient.SqlCommand(  ↙
    "spClearExpiredSessions", sqlConn);
            sqlCmd.CommandType = System.Data.CommandType.StoredProcedure;

            // Execute the stored procedure
            sqlCmd.ExecuteScalar();
        }

        /// <summary>
        /// Verifies SQL input to protect against injection attacks.
        /// </summary>
        /// <param name="sql">The SQL variable we'll be cleaning</param>
        private String CleanSQL(String sql)
        {
            // Make sure our SQL string is alphanumeric
            if (reg.IsMatch(sql))
            {
                return sql;
            }
            else {
                return "";
            }
        }
    }
}
```