

March 10th

08

Creating MD5 Hashes Without a Plugin

Tom Parry

Prospect IT Consulting, Inc. venture@compmore.net

Java for Servoy Developers

Community support for all things Java and Servoy

<http://java-servoy.blogspot.com>

This is a Blog Post!

This is a PDF version of a blog post located at <http://java-servoy.blogspot.com>. It is not a formal whitepaper or other similar guide and is intended to be a casual writing. Apart from styling and formatting differences, and the possible addition of graphics it is typically identical to its online counterpart.

Sometimes things just look better in print 😊

Creating MD5 Hashes Without a Plugin

Every had the need to have a secure password on your system for logins? Well, if one had to use the MD5 hash mechanism (MD5 refers to a message digest) from a Servoy application one would be hard pressed to do it properly. Java has a Message Digest class that is ideal to use from javascript. There are a few tricks to employ in order to make the output compliant with the algorithm.

The typical processing is as follows:

1. In an administration setup form get the plain text password from the user.
2. Create a MD5 hash from the plain text input.
3. Store the resultant hash in the database in a column of a table that has limited access.
4. In a login form get the user to enter his password (using whatever masking tricks you have to not echo to the screen text field of course!).
5. Create another MD5 hash from this password.
6. Compare the two hashes – the one from the table in the db and the other from the just created password.

Realize of course that MD5 is a simple hash and is able to be broken and of course it is not such a clever idea to store the password hash in a very open database. However for those whose needs are to provide a basic level of password protection this might do the trick.

Remember that there are web sites that are able to do a lookup in their database of the key to a hash that was previously entered as part of their "service". So maybe try to enforce "strong" password rules so that it might be a little more difficult to reverse.

That said the following code is a generic method in a module of my Servoy solutions. I have not made it very efficient so that you can see some of the intermediate steps more clearly – especially if you use the debugger to step through and see the values changing.

As always, let me know if this is defective in some manner so we can all benefit. (Note that the try...catch appears to be necessary but I have not checked it).

The most difficult thing to do here is to convert the original text string input to a java.lang.Byte array which is what the message digest class desires as input. Firstly the input string is "sliced" apart into one character per array element. Then each of those elements is converted to a java.lang.Byte ending up with an array of Byte instances.

The actual "crunching" of the hash is really just the two lines:

```
var md = Packages.java.security.MessageDigest.getInstance("MD5");  
  
var md5_byteArray = md.digest(in_array);
```

Note that the argument to the getInstance is "MD5". Another popular argument is "SHA". For more information I suggest reading:

<http://java.sun.com/javase/6/docs/api/java/security/MessageDigest.html>

```

2008.March.05
/*
  Global method to calculate a MD5 hash from a string.
4   IN: arguments[0] String of text to convert.
5   OUT: String equivalent to the MD5 hash.
6 */
7
8   if (!arguments || !arguments[0] )
9     return null;
10  //get an array of bytes from the input string
11  var in_array = Array(arguments[0].length);
12  var in_char;
13  for (var i = 0; i < arguments[0].length; i++) {
14    //one character - force to string type by the plus ''
15    in_char = arguments[0].slice(i,i+1) + '';
16    in_array[i] = in_char.charCodeAt(0);
17  }
18  //convert to a Byte array
19  var inByteArray = new Array(in_array.length);
20  for (i = 0; i < in_array.length; i++) {
21    inByteArray[i] = Packages.java.lang.Byte( in_array[i] );
22  }
23  //instantiate the Message Digest
24  //static method - do not use 'new'
25  var md = Packages.java.security.MessageDigest.getInstance("MD5");
26  try {
27    var md5_byteArray = md.digest(in_array);
28  } catch (e) {
29    application.output("Exception make_md5");
30    return null;
31  }
32  var md5_hash = ''; //empty string type
33  var md5_array = new Array(md5_byteArray.length); //
34  for(i = 0; i < md5_byteArray.length; i++) {
35    //get hex code for each
36    md5_array[i] = Packages.java.lang.Integer.toHexString(md5_byteArray[i]);
37    if (md5_array[i].length > 2) {
38      //take right most 2 chars
39      md5_array[i] = md5_array[i].substr(md5_array[i].length - 2, 2);
40    }
41
42    if ((md5_byteArray[i] >= 0 ) && (md5_byteArray[i] <= 15)) {
43      //pre-pend a '0' since the hex to string fails to do so - typical to forget!
44      md5_array[i] = '0' + md5_array[i];
45    }
46    md5_hash += md5_array[i]; //concatenate
47  }
48  //application.output("result = " + md5_hash); //for debugging usage only
49  return md5_hash;

```

After getting the result in the var md5_byteArray it has to be transformed in to the string of hex characters. Another tricky bit here! Luckily java has a clever way of providing a method called `java.lang.Integer.toHexString` to convert the number in the Byte array element (an Integer) to a string of the hexadecimal equivalent of the integer value. But then we realize that some numbers were negative but the `toHexString` only deals in unsigned integers. Not to worry! We just check if the output string has more than two characters and keep the rightmost two!

The next problem we face (and some programmers fail to recognize that this is a problem) is that a proper hash result for hexadecimal numbers 0-15 must always be two characters long. So for these cases we just prepend a '0' to the hex string value. (Of course one could create a "variant" of MD5 by deciding not to do this but then it is not compliant with nor will it have the same result if your backend database has an MD5 function that you could use).

As always, good luck and let me know if there are problems or improvements.

Tom Parry

Prospect IT Consulting Inc

venture@compmore.net