ECM3408 - CA1: Skydrive

62007094

February 15, 2015

1 SQLite Database Creation

ew The following commands can be used to create a table called `documents`

```
$ sqlite3 skydrive.sqlite3
sqlite > CREATE TABLE documents
    ...> ( id INT PRIMARY KEY
    ...> , name VARCHAR
    ...> , message VARCHAR ); # sqlite does not impose length-limits
       on fields, so none is given.
.quit
```

1.1 Testing

Script output of creating a database as described above, storing an unencrypted message and retrieving it:

```
bash-3.2$ sqlite3 skydrive.sqlite3
SQLite version 3.8.5 2014-08-15 22:37:57
Enter ".help" for usage hints.
sqlite> CREATE TABLE documents
   ...> ( id INT PRIMARY KEY
   ...> , name VARCHAR
   ...> , message VARCHAR );
sqlite> INSERT INTO documents VALUES
   ...> (1, "Name", "Message");
sqlite> SELECT * FROM documents;
1|Name|Message
sqlite>
```

2 The Ruby Code

```
#!/usr/bin/env ruby -w
require "sqlite3"
require "webrick"
DATABASE = "skydrive.sqlite3"

# -- MODELS --

##
# Model for the Index: Gets a list of all documents and their IDs for the
    view
def model_index()
```

```
11      db  = SQLite3::Database.new( DATABASE )
12      qry = "SELECT id, name FROM documents;"
13      hash = db.execute( qry )
14      db.close
15      return hash
16  end
17
18  ##
19  # Model for showing an individual message
20  # Gets the message via id, and decrypts with the cipher given
21  def model_show(id, shift)
22      db  = SQLite3::Database.new( DATABASE )
23      qry = "SELECT message FROM documents "                              +
24                "WHERE id = \"#{id}\";"
25      val = db.get_first_value( qry )
26      cipher = Caesar.new shift
27      message = cipher.decrypt(val)
28      db.close
29      return message
30  end
31
32  ##
33  # Model to update an entry
34  # Depending on whether an entry has just been updated, an entry has just
        been
35  # selected to be updated or no entry has been selected, yet, the
        appropriate
36  # data is processed
37
38  def model_update(show=false, update=false, id, message, shift)
39      db  = SQLite3::Database.new( DATABASE )
40      qry = "SELECT id, name FROM documents;"
41      hash = db.execute( qry )
42
43      cipher = Caesar.new shift.to_i
44
45      ##
46      # load an entry to be edited
47      if show
48        qry = "SELECT message FROM documents "                            +
49        "WHERE id = \"#{id}\";"
50        val = db.get_first_value( qry )
51        message_dec = cipher.decrypt(val)
52        db.close
53        return false, hash, message_dec, id
54
55      ##
56      # update an entry with the text already entered
57      elsif update
58        message_enc = cipher.encrypt(message)
59        puts message_enc
60        puts shift
61        puts id
```

```
62        qry= "UPDATE documents SET message=\"#{message_enc}\""        +
63             "WHERE id=\"#{id}\";"
64        db.execute( qry )
65        return true, hash
66
67      ##
68      # just show the entries available for edit
69      else
70        return false, hash
71      end
72  end
73
74  ##
75  # Model for creating a new entry. Depending on whether data has already
        been
76  # entered or not, the appropriate data is processed
77
78  def model_new(process=false, message, name, shift)
79      ##
80      # if data has been entered to save
81      if process
82        cipher = Caesar.new shift.to_i
83        message_enc = cipher.encrypt(message)
84
85        db  = SQLite3::Database.new( DATABASE )
86        qry = "SELECT id FROM documents ORDER BY id DESC LIMIT 1;"
87        id = db.execute( qry ).join.to_i + 1
88        qry = "INSERT INTO documents VALUES"                          +
89            "(#{id}, \"#{name}\", \"#{message_enc}\");"
90        db.execute( qry )
91        db.close
92        return true
93      end
94      return false
95  end
96
97
98  ##
99  # Model for destroying an existing entry
100
101 def model_destroy(process=false, id)
102     db  = SQLite3::Database.new( DATABASE )
103
104     ##
105     # if an entry has been selected to be destroyed
106     if process
107       qry = "DELETE FROM documents WHERE id=#{id};"
108       db.execute (qry)
109       qry = "SELECT id, name FROM documents;"
110       hash = db.execute( qry )
111       return true, hash
112     end
113
```

```
114     ##
115     # load a list of entries for selection
116     qry = "SELECT id, name FROM documents;"
117     hash = db.execute( qry )
118
119     db.close
120     return false, hash
121 end
122
123 # -- VIEWS --
124
125 def view_show(val)
126     "<html>"                                                    +
127     "  <body>"                                                  +
128     "    <p>" + val.to_s + "</p>"                               +
129     "  </body>"                                                 +
130     "</html>"
131 end
132
133 def view_index(vals)
134     output = "<html>"                                           +
135     "  <body>"                                                  +
136     "    <form action=\"http://localhost:3000/show\""           +
137     "    method=\"GET\">"                                       +
138     "      <select name=\"id\">"
139
140     ##
141     # for each value, show name and have id as form-value
142     vals.each do |key, value|
143           output << "<option value=\"#{key}\">#{value}</option><br \>"
144     end
145
146     output << "</select>"                                       +
147     "      <input name=\"shift\" value=\"Enter shift\"/>"       +
148     "      <input type=\"Submit\"/>"                            +
149     "    </form>"                                               +
150     "  </body>"                                                 +
151     "</html>"
152 end
153
154 def view_new(success=false)
155     output = "<html>"                                           +
156     "  <body>"
157
158     if success
159         output << "<p>Message encrypted and added</p>"
160     end
161
162     output << "    <form action=\"http://localhost:3000/new\""  +
163     "    method=\"GET\">"                                       +
164     "      <input name=\"shift\" value=\"Enter Shift\"/>"       +
165     "      <input name=\"name\" value=\"Enter Name\"/>"         +
166     "      <input name=\"message\" value=\"Enter Message\"/>"   +
```

```
167       "       <input type=\"Submit\"/>"                                +
168       "     </form>"                                                   +
169       "   </body>"                                                     +
170       "</html>"
171
172       return output
173   end
174
175   def view_destroy(deleted=false, vals)
176       output = "<html>"                                                +
177       "   <body>"
178
179       if deleted
180           output << "<p>Message deleted</p>"
181       end
182
183       output << "     <form action=\"http://localhost:3000/destroy\"" +
184       "     method=\"GET\">"                                           +
185       "       <select name=\"id\">"
186
187       ##
188       # for each value, show name and have id as form-value
189       vals.each do |key, value|
190         output << "<option value=\"#{key}\">#{value}</option><br \>"
191       end
192
193       output << "</select>"                                            +
194       "       <input type=\"Submit\"/>"                                +
195       "     </form>"                                                   +
196       "   </body>"                                                     +
197       "</html>"
198
199       return output
200   end
201
202   def view_update(updated, vals, msg, id, shift)
203       output = "<html>"                                                +
204       "   <body>"
205
206       if updated
207           output << "<p>Message updated</p>"
208       end
209
210       ##
211       # if the user requested to update an entry, the model will provide
              the
212       # decrypted message to edit. This is checking if a message has been
213       # transmitted and displays the edit form if that is the case
214       if defined? msg
215         output << "     <form action=\"http://localhost:3000/update\""+
216         "     method=\"GET\">"                                         +
217         "       <input type=\"hidden\" name=\"id\" value=\"#{id}\"/>" +
```

```
218         "       <input type=\"hidden\" name=\"shift\" value=\"#{shift}\"/>"
              +
219         "       <input name=\"message\" value=\"#{msg}\"/>"            +
220         "       <input type=\"Submit\"/>"                              +
221         "     </form>"
222       end
223
224       output << "    <form action=\"http://localhost:3000/update\""  +
225       "     method=\"GET\">"                                          +
226       "       <select name=\"id\">"
227
228       ##
229       # show all entries that can be edited
230       vals.each do |key, value|
231         output << "<option value=\"#{key}\">#{value}</option><br \>"
232       end
233
234       output << "</select>"                                           +
235       "       <input name=\"shift\" value=\"Enter shift\"/>"         +
236       "       <input type=\"Submit\"/>"                              +
237       "     </form>"                                                 +
238       "   </body>"                                                   +
239       "</html>"
240
241       return output
242   end
243
244
245   # -- CONTROLLER --
246
247   class Controller < WEBrick::HTTPServlet::AbstractServlet
248       def do_GET ( req, rsp )
249         ##
250         # Decide on which MV by analysing the request
251         case req.path
252           ##
253           # Index: Overview of all messaged
254           when "/index"
255             rsp.status = 200
256             rsp.content_type = "text/html"
257             rsp.body = view_index( model_index() )
258
259           ##
260           # Add new message
261           when "/new"
262             message = req.query[ "message" ] || ""
263             name = req.query[ "name" ] || ""
264             shift = req.query[ "shift" ] || ""
265
266             ##
267             # Check if something has been submitted for processing
268             if message.length == 0 || name.length == 0 || shift.length == 0
269               process = false
```

```
270             else
271               process = true
272             end
273
274             rsp.status = 200
275             rsp.content_type = "text/html"
276             rsp.body = view_new( model_new(process, message, name, shift) )
277
278           ##
279           # Showing an entry
280           when "/show"
281             id = req.query[ "id" ] || ""
282             shift = req.query[ "shift" ] || ""
283             rsp.status = 200
284             rsp.content_type= "text/html"
285             rsp.body = view_show( model_show(id, shift.to_i) )
286
287           ##
288           # Destroying an entry
289           when "/destroy"
290             id = req.query[ "id" ] || ""
291
292             ##
293             # Check if an entry has been submitted to be deleted
294             if id.length == 0
295               process = false
296             else
297               process = true
298             end
299
300             rsp.status = 200
301             rsp.content_type = "text/html"
302             success, vals = model_destroy(process, id)
303             rsp.body = view_destroy(success, vals )
304
305           ##
306           # Update existing entry
307           when "/update"
308             message = req.query[ "message" ] || ""
309             id = req.query[ "id" ] || ""
310             shift = req.query[ "shift" ] || ""
311
312             ##
313             # If message already submitted to upgrade old one, call
                  appropriately
314             if message.length > 0
315               update = true
316               show = false
317             elsif id.length > 0 && shift.length > 0
318               update = false
319               show = true
320             end
321
```

```
322             rsp.status = 200
323             rsp.content_type = "text/html"
324
325             updated, vals, msg, id = model_update(show, update,
326                                                   id, message, shift)
327
328             rsp.body = view_update(updated, vals, msg, id, shift)
329         end
330       end
331 end
332
333 # -- SUPPORT CLASSES --
334
335 ##
336 # Caesar: Providing the encryption scheme and functions to encrypt
337 class Caesar
338
339   ##
340   # Constructor - takes amount of shift as arguemtn, and uses an alphabet
          of all
341   # letter (capital or not), numbers and whitespace by default
342   def initialize(shift, alphabet = (('a'..'z').to_a + ('A'..'Z')
343                                     .to_a + ('0'..'9').to_a + [' '])
344                                     .join)
345     ##
346     # Put alphabet in array and rotate array by amount of shift if de-/
          encrypt
347     # requested
348     chars = alphabet.chars.to_a
349     @encrypter = Hash[chars.zip(chars.rotate(shift))]
350     @decrypter = Hash[chars.zip(chars.rotate(-shift))]
351   end
352
353   def encrypt(string)
354     @encrypter.values_at(*string.chars).join
355   end
356
357   def decrypt(string)
358     @decrypter.values_at(*string.chars).join
359   end
360 end
361
362 # -- RUNTIME --
363
364 server = WEBrick::HTTPServer.new( :Port => 3000 )
365    server.mount( "/", Controller )
366    server.start
```

3 Testing

Unit	Process	Expected Outcome	Actual Outcome
Index	Call /index	List of all saved messages	As Expected
Index/Show	Select a message on /index, enter the shift and submit	Opens /show with decrypted message. Shows only scrambled message with the wrong key provided	As Expected
New	Call /new	Option to enter new message with name, shift and message body	As Expected
Add New	Submit new entry and have it written to the database	Submit form at /new with filled out forms andante confirmation of entry been written, and have it display on /index and read it on /show with the shift originally provided. Scrambled output if providing any other shift key	As expected
Destroy	Open /destroy and delete an entry	Get presented with a list of entries, select one for deletion, get confirmation, and not be able to access it anymore on /index	As Expected
Update	Open /update, select an entry to be updated, enter the shift key, edit and save	As process and then check on /index and subsequently on /show whether the change was successful	As Expected

All components tested for functionality and purposes succesfully.