

Oct 19, 15 11:11

Model.py

Page 1/1

```

1 from google.appengine.api import users
2 from google.appengine.ext import ndb
3
4 """ This class has the DTOs used to retrieve and store data from the
5 DB and also has the keys to obtain them
6
7 Store all the events with the same key just because google said it is
8 necessary
9 """
10
11 DEFAULT_EVENT_BOX = '-'
12
13 def events_key():
14     return ndb.Key('global', DEFAULT_EVENT_BOX)
15
16
17 def event_guests_key(event_name):
18     return ndb.Key('Event', event_name)
19
20
21
22 class Event(ndb.Model):
23     name = ndb.StringProperty()
24     capacity = ndb.StringProperty()
25     # Amount of Guests in the Event
26     guests = ndb.StringProperty(indexed=False)
27
28
29 class Guest(ndb.Model):
30     email = ndb.StringProperty(indexed=True)
31     name = ndb.StringProperty(indexed=False)
32     surname = ndb.StringProperty(indexed=False)
33     company = ndb.StringProperty(indexed=False)

```

Oct 19, 15 11:29

index.py

Page 1/4

```

1 <!DOCTYPE html>
2 {% autoescape true %}
3 <html>
4     <head>
5         <link type="text/css" rel="stylesheet" href="/stylesheets/main.css" />
6         <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
7     </head>
8     <body onload="pageLoaded()">
9         <h1>Towers' Event Hosting Web Page</h1>
10        <div id="homerWebPage" class="toshow" float="right">
11            <!--
12            
13            
14            
15            
16            
17            -->
18        </div>
19
20        <form id="eventCreationForm" action="/event_creation" method="post">
21            <h2>Create a new Event:</h2>
22            <br>
23            <label>Event Name:</label>
24            <input type="text" id="eventCreationName" name="eventName">
25            <br>
26
27            <label>Event Capacity:</label>
28            <input type="text" id="eventCreationCapacity" name="eventCapacity">
29            <br>
30
31            <input type="submit" value="Add Event">
32        </form>
33
34        <hr>
35
36        <form id="eventManagementForm" action="/event_management" method="post">
37            <label>Choose an event: </label>
38            <select id="eventDropdown" onchange="updateSelectedEventData()">
39                <option value="" disabled selected style="display:none;">--</option>
40                {% for event in events %}
41                <option value="{ {event.name} }">{ {event.name} }</option>
42                {% endfor %}
43            </select>
44            <br>
45
46            <label id="eventCapacity">Event Capacity: 0</label>
47            <br>
48            <label id="eventGuests">Suscribed Guests: 0</label>
49        </form>
50
51        <button id="deleteEvent" onclick="deleteEvent()">
52            Delete Actual Event
53        </button>
54
55        {% for guests in guests_list %}
56        <div id="table{{ {guests[0]} }}">
57            <h3> {{ {guests[0]} }} Guests List</h3>
58            <table class="cTable">
59                <tr>
60                    <th> Name </th>
61                    <th> Surname </th>
62                    <th> E-Mail </th>
63                    <th> Company </th>
64                </tr>
65                {% for guest in guests[1] %}
66                <tr>
67                    <td> {{ {guest.name} }} </td>
68                    <td> {{ {guest.surname} }} </td>
69                    <td> {{ {guest.email} }} </td>
70                    <td> {{ {guest.company} }} </td>
71                </tr>
72                {% endfor %}
73            </table>

```

Oct 19, 15 11:29

index.py

Page 2/4

```

74 </div>
75 {% endfor %}
76
77 <hr>
78 <form id="guestsForm" action="/add_guest" method="post">
79 <h3 id="guestsFormTitle"></h3>
80 <label>Name:</label>
81 <input type="text" id="guestName" name="guestName">
82 <br>
83
84 <label>Surname:</label>
85 <input type="text" id="guestSurname" name="guestSurname">
86 <br>
87
88 <label>E-Mail:</label>
89 <input type="text" id="guestEmail" name="guestEmail">
90 <br>
91
92 <label>Company:</label>
93 <input type="text" id="guestCompany" name="guestCompany">
94 <br>
95
96 <input type="submit" value="Join">
97 </form>
98 <hr>
99
100 <!-- JQuery scripts -->
101 <script>
102 $("#guestCompany").keydown(function(event) {
103     if (event.keyCode == 118) {
104         $("#homerWebPage").show();
105     }
106 });
107
108 $("#guestsForm").submit(function(event) {
109     event.preventDefault();
110
111     if ($("#guestName").val() == "") {
112         alert("Guest Name can't be empty");
113         return;
114     }
115     if ($("#guestSurname").val() == "") {
116         alert("Guest Surname can't be empty");
117         return;
118     }
119     if ($("#guestEmail").val() == "") {
120         alert("Guest Email can't be empty");
121         return;
122     }
123     if ($("#guestCompany").val() == "") {
124         alert("Guest Company can't be empty");
125         return;
126     }
127
128     var url = $(this).attr("action");
129     var actualEvent = $("#selectEventDropdown").val();
130     var data = $(this).serializeArray();
131     data.push( {name: "actualEvent", value: actualEvent} );
132     data.push( {name: "checkDuplicates", value: "true"} );
133     $.post(url, data,
134     function(data) {
135         if (data.add == "true") {
136             alert("Guest was successfully joined to the event.");
137         }
138         else if (data.add == "false") {
139             alert("Event is full. Could not add guest.");
140         }
141         else if (data.add == "duplicated") {
142             alert("Guest already subscribed to event.");
143         }
144         window.location.reload(true);
145     }, "json");
146 });

```

Oct 19, 15 11:29

index.py

Page 3/4

```

147
148     $("#eventCreationForm").submit(function(event) {
149         var name = $("#eventCreationName").val();
150         if (name == "") {
151             alert("Event name must not be empty");
152             event.preventDefault();
153             return;
154         }
155
156         var capacity = $("#eventCreationCapacity").val();
157         if (capacity == "") {
158             alert("Event capacity must not be empty");
159             event.preventDefault();
160             return;
161         }
162
163         if ($.isNumeric(capacity) != true) {
164             alert("Event capacity must be a number");
165             event.preventDefault();
166             return;
167         }
168     });
169 </script>
170
171 <!-- Plain javascript functions -->
172 <script type="text/javascript">
173     function pageLoaded() {
174         $("#guestsForm").hide();
175         $("#deleteEvent").hide();
176         $("#homerWebPage").hide();
177
178         {%for event in events %}
179             $("#table{event.name}").hide();
180         {% endfor %}
181     }
182
183     function deleteEvent() {
184         var e = document.getElementById("eventDropdown");
185         var value = e.options[e.selectedIndex].value;
186
187         var actualEvent = {event : value};
188         $.post("/event_removal", actualEvent, function(data) {
189             if (data.deleted == "true") {
190                 alert("Event was successfully deleted.");
191             }
192             else {
193                 alert("Event could not be deleted.");
194             }
195             window.location.reload(true);
196         }, "json");
197     }
198
199     function updateSelectedEventData() {
200         var e = document.getElementById("eventDropdown");
201         var value = e.options[e.selectedIndex].value;
202
203         {% for event in events %}
204             $("#table{event.name}").hide();
205         {% endfor %}
206
207         if (value != "-") {
208             $("#guestsForm").show();
209             document.getElementById("guestsFormTitle").innerHTML =
210                 value + " Event Subscription";
211             $("#table" + value).show();
212             $("#deleteEvent").show();
213         }
214
215         {% for event in events %}
216         if (value == "{{event.name}}") {
217             document.getElementById("eventCapacity").innerHTML
218                 = "Event Capacity: " + "{{event.capacity}}";
219             document.getElementById("eventGuests").innerHTML

```

Oct 19, 15 11:29

index.py

Page 4/4

```

220         = 'Suscribed Guests: ' + "{{event.guests}}";
221     }
222     {% endfor %}
223
224     return;
225 }
226 </script>
227
228 </body>
229 </html>
230 {% endautoescape %}

```

Oct 19, 15 11:11

EventsCreation.py

Page 1/1

```

1  from Model.Model import *
2  from Model.Model import Event
3
4  import webapp2
5  import urllib
6
7
8  class EventsCreation(webapp2.RequestHandler):
9      def post(self):
10         # Get all the events to check if the event already exists
11         name = self.request.get('eventName')
12         capacity = int(self.request.get('eventCapacity'))
13
14         # Upload a new event if this name and capacity are valid
15         if name:
16             event = Event(parent=events_key())
17             event.name = name
18             event.capacity = str(capacity)
19             event.guests = str(0)
20             event.put()
21
22         query_params = {'event_name': name}
23         self.redirect('/?' + urllib.urlencode(query_params))

```

Oct 19, 15 15:37

EventRemoval.py

Page 1/1

```

1 from google.appengine.ext import ndb
2 from Model.Model import *
3
4 import webapp2
5 import json
6 import logging
7
8 class EventRemoval(webapp2.RequestHandler):
9     @ndb.transactional(xg=True)
10     def delete_event(self, event_name):
11         events_query = Event.query(ancestor=events_key())
12         events = events_query.fetch()
13
14         found = False
15         for event in events:
16             if event.name == event_name:
17                 logging.warning("DELETING EVENT " + event_name)
18                 event.key.delete()
19                 found = True
20
21         if not found:
22             return False
23
24         # Delete all the guests related with this event
25         guests_query = Guest.query(ancestor=event_guests_key(event_name))
26         guests = guests_query.fetch()
27
28         logging.warning("DELETING GUESTS OF EVENT" + event_name)
29         for guest in guests:
30             guest.key.delete()
31
32         return True
33
34 def post(self):
35     actual_event = self.request.get('event')
36     logging.debug("Event to remove: " + actual_event)
37
38     if self.delete_event(actual_event):
39         self.response.headers['Content-Type'] = "application/json"
40         self.response.out.write(json.dumps({'deleted': 'true'}))
41     else:
42         self.response.headers['Content-Type'] = "application/json"
43         self.response.out.write(json.dumps({'deleted': 'false'}))
44

```

Oct 19, 15 11:32

AddGuest.py

Page 1/1

```

1 from google.appengine.ext import ndb
2 from Model.Model import *
3
4 import webapp2
5 import json
6 import logging
7
8 class AddGuest(webapp2.RequestHandler):
9     @ndb.transactional(xg=True)
10     def increment_guest_count(self, event_name):
11         # Get the event
12         events_query = Event.query(Event.name == event_name,
13                                   ancestor=events_key())
14         event = events_query.get()
15
16         logging.debug(
17             "Event: " + event_name +
18             " - Capacity: " + event.capacity)
19         logging.debug(
20             "Event: " + event_name +
21             " - Guests: " + event.guests)
22         if int(event.capacity) > int(event.guests):
23             logging.debug(
24                 "Event " + event_name +
25                 ": There is capacity")
26             amount_guests = int(event.guests) + 1
27             event.guests = str(amount_guests)
28             event.put()
29             return True
30         else:
31             logging.debug(
32                 "Event " + event_name +
33                 ": There ISN'T capacity")
34             return False
35
36     def guest_exists(self, guest_email, event):
37         guest_query = Guest.query(Guest.email == guest_email,
38                                   ancestor=event_guests_key(event))
39         guest = guest_query.get()
40         if guest:
41             logging.warning(
42                 "Guest with email " + guest_email
43                 + " already exists. Dropping request")
44             return True
45         return False
46
47     def post(self):
48         event_name = self.request.get('actualEvent')
49         check_duplicates = self.request.get('checkDuplicates')
50
51         if check_duplicates == "true" and \
52            self.guest_exists(self.request.get('guestEmail'), event_name):
53             self.response.headers['Content-Type'] = "application/json"
54             self.response.out.write(json.dumps({'add': 'duplicated'}))
55             return
56
57         if not self.increment_guest_count(event_name):
58             self.response.headers['Content-Type'] = "text/plain"
59             self.response.out.write(json.dumps({'add': 'false'}))
60             return
61         else:
62             # Add the guest
63             guest = Guest(parent=event_guests_key(event_name))
64             guest.name = self.request.get('guestName')
65             guest.surname = self.request.get('guestSurname')
66             guest.email = self.request.get('guestEmail')
67             guest.company = self.request.get('guestCompany')
68             guest.put()
69
70             self.response.headers['Content-Type'] = "application/json"
71             self.response.out.write(json.dumps({'add': 'true'}))
72

```

Oct 19, 15 11:11

application.py

Page 1/1

```

1 import os
2 from Model.Model import *
3 from Handlers.AddGuest import AddGuest
4 from Handlers.EventsCreation import EventsCreation
5 from Handlers.EventRemoval import EventRemoval
6
7 import jinja2
8 import webapp2
9
10 JINJA_ENVIRONMENT = jinja2.Environment(
11     loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
12     extensions=['jinja2.ext.autoescape'],
13     autoescape=True)
14
15
16 class MainPage(webapp2.RequestHandler):
17     def get(self):
18         events_query = Event.query(ancestor=events_key())
19         events = events_query.fetch()
20
21         guest_list = []
22         for event in events:
23             guest_query = Guest.query(ancestor=event_guests_key(event.name))
24             guests_by_event = guest_query.fetch()
25             guest_list.append((event.name, guests_by_event))
26
27         template_values = {
28             'events': events,
29             'guests_list': guest_list
30         }
31
32         template = JINJA_ENVIRONMENT.get_template('index.html')
33         self.response.write(template.render(template_values))
34
35
36 app = webapp2.WSGIApplication([
37     ('/', MainPage),
38     ('/updater', MainPage),
39     ('/event_creation', EventsCreation),
40     ('/event_removal', EventRemoval),
41     ('/add_guest', AddGuest),
42 ], debug=True)

```

Oct 19, 15 15:43

Table of Content

Page 1/1

| | | | | | |
|---|--------------------------|--------|------|--------------|----------------|
| 1 | Table of Contents | | | | |
| 2 | 1 Model.py..... | sheets | 1 to | 1 (1) pages | 1- 1 34 lines |
| 3 | 2 __init__.py..... | sheets | 1 to | 1 (1) pages | 1- 1 1 lines |
| 4 | 3 index.py..... | sheets | 1 to | 3 (3) pages | 2- 5 231 lines |
| 5 | 4 __init__.py..... | sheets | 3 to | 3 (1) pages | 5- 5 1 lines |
| 6 | 5 EventsCreation.py... | sheets | 3 to | 3 (1) pages | 6- 6 24 lines |
| 7 | 6 EventRemoval.py..... | sheets | 4 to | 4 (1) pages | 7- 7 45 lines |
| 8 | 7 AddGuest.py..... | sheets | 4 to | 4 (1) pages | 8- 8 73 lines |
| 9 | 8 application.py..... | sheets | 5 to | 5 (1) pages | 9- 9 43 lines |