Oct 19, 15 11:11 **Model.py** Page 1/1

```python
from google.appengine.api import users
from google.appengine.ext import ndb

""" This class has the DTOs used to retrieve and store data from the
    DB and also has the keys to obtain them

    Store all the events with the same key just because google said it is
    necessary
"""

DEFAULT_EVENT_BOX = '-'


def events_key():
    return ndb.Key('global', DEFAULT_EVENT_BOX)


def event_guests_key(event_name):
    return ndb.Key('Event', event_name)


class Event(ndb.Model):
    name = ndb.StringProperty()
    capacity = ndb.StringProperty()
    # Amount of Guests in the Event
    guests = ndb.StringProperty(indexed=False)


class Guest(ndb.Model):
    email = ndb.StringProperty(indexed=True)
    name = ndb.StringProperty(indexed=False)
    surname = ndb.StringProperty(indexed=False)
    company = ndb.StringProperty(indexed=False)
```

Oct 19, 15 11:29 **index.py** Page 1/4

```
<!DOCTYPE html>
{% autoescape true %}
<html>
  <head>
  <link type="text/css" rel="stylesheet" href="/stylesheets/main.css" />
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
  </head>
  <body onload="pageLoaded()">
  <h1>Towers' Event Hosting Web Page</h1>
 <div id="homerWebPage" class="toshow" float="right">
  <!--<img src="http://www.homerswebpage.com/flash/tostadora_100px1.gif"/>
  <img src="http://www.homerswebpage.com/flash/boca_90px1.gif"/>
  <img src="http://www.homerswebpage.com/flash/gusano_60px1.gif"/>
  <img src="https://38.media.tumblr.com/tumblr_m674jlpyPT1ry8fquo1_250.gif"/>
  <img src="http://www.homerswebpage.com/flash/campana_60px1.gif"/>
  <img src="http://www.homerswebpage.com/flash/reloj_60px1.gif"/>
  <img src="http://www.homerswebpage.com/flash/tostadora_100px1.gif"/>-->
 </div>

 <form id="eventCreationForm" action="/event_creation" method="post">
  <h2>Create a new Event:</h2>
  <br>
  <label>Event Name:</label>
  <input type="text" id ="eventCreationName" name="eventName">
  <br>

  <label>Event Capacity:</label>
  <input type="text" id ="eventCreationCapacity" name="eventCapacity">
  <br>

  <input type="submit" value="Add Event">
 </form>

 <hr>

 <form id="eventManagementForm" action="/event_management" method="post">
  <label>Choose an event: </label>
  <select id="eventDropdown" onchange="updateSelectedEventData()">
  <option value="" disabled selected style="display:none;">-</option>
  {% for event in events %}
  <option value="{{event.name}}">{{event.name}}</option>
  {% endfor %}
  </select>
  <br>

  <label id="eventCapacity">Event Capacity: 0</label>
  <br>
  <label id="eventGuests">Suscribed Guests: 0</label>
 </form>

 <button id ="deleteEvent" onclick="deleteEvent()">
  Delete Actual Event
 </button>

 {% for guests in guests_list %}
 <div id="table{{guests[0]}}">
  <h3> {{guests[0]}} Guests List</h3>
  <table class="cTable">
   <tr>
    <th> Name </th>
    <th> Surname </th>
    <th> E-Mail </th>
    <th> Company </th>
   </tr>
   {% for guest in guests[1] %}
   <tr>
    <td> {{ guest.name }} </td>
    <td> {{ guest.surname }} </td>
    <td> {{ guest.email }} </td>
    <td> {{ guest.company }} </td>
   </tr>
   {% endfor %}
  </table>
```

```
</div>
{% endfor %}

<hr>
<form id="guestsForm" action="/add_guest" method="post">
<h3 id ="guestsFormTitle"></h3>
<label>Name:</label>
<input type="text" id="guestName" name="guestName">
<br>

<label>Surname:</label>
<input type="text" id="guestSurname" name="guestSurname">
<br>

<label>E−Mail:</label>
<input type="text" id="guestEmail" name="guestEmail">
<br>

<label>Company:</label>
<input type="text" id="guestCompany" name="guestCompany">
<br>

<input type="submit" value="Join">
</form>
<hr>

<!−− JQuery scripts −−>
<script>
 $("#guestCompany").keydown(function(event) {
  if (event.keyCode == 118) {
    $("#homerWebPage").show();
  }
 });

 $("#guestsForm").submit(function(event) {
  event.preventDefault();

  if ($("#guestName").val() == "") {
   alert("Guest Name can't be empty");
   return;
  }
  if ($("#guestSurname").val() == "") {
        alert("Guest Surname can't be empty");
        return;
      }
      if ($("#guestEmail").val() ≡ "") {
        alert("Guest Email can't be empty");
        return;
      }
      if ($("#guestCompany").val() ≡ "") {
        alert("Guest Company can't be empty");
        return;
      }

      var url = $(this).attr("action");
      var actualEvent = $("select#eventDropdown").val();
      var data = $(this).serializeArray();
      data.push( {name: "actualEvent", value: actualEvent} );
      data.push( {name: "checkDuplicates", value: "true"} );
      $.post(url, data,
        function(data) {
          if (data.add ≡ "true") {
            alert("Guest was succesfully joined to the event.");
          }
          else if (data.add ≡ "false") {
            alert("Event is full. Could not add guest.");
          }
          else if (data.add ≡ "duplicated") {
            alert("Guest already suscribed to event.");
          }
          window.location.reload(true);
        }, "json");
    });
```



```

    $("#eventCreationForm").submit(function(event) {
      var name = $("#eventCreationName").val();
      if (name ≡ "") {
        alert("Event name must not be empty");
        event.preventDefault();
        return;
      }

      var capacity = $("#eventCreationCapacity").val();
      if (capacity ≡ "") {
        alert("Event capacity must not be empty");
        event.preventDefault();
        return;
      }

      if ($.isNumeric(capacity) ≠ true) {
        alert("Event capacity must be a number");
        event.preventDefault();
        return;
      }
    });
  </script>

  <!−− Plain javascript functions −−>
  <script type="text/javascript">
    function pageLoaded() {
      $("#guestsForm").hide();
      $("#deleteEvent").hide();
      $("#homerWebPage").hide();

      {%for event in events %}
        $("#table{{event.name}}").hide();
      {% endfor %}
    }

    function deleteEvent() {
      var e = document.getElementById("eventDropdown");
      var value = e.options[e.selectedIndex].value;

      var actualEvent = {event : value};
      $.post("/event_removal", actualEvent, function(data) {
          if (data.deleted ≡ "true") {
            alert("Event was succesfully deleted.");
          }
          else {
            alert("Event could not be deleted.");
          }
          window.location.reload(true);
        }, "json");
    }

    function updateSelectedEventData() {
      var e = document.getElementById("eventDropdown");
      var value = e.options[e.selectedIndex].value;

      {% for event in events %}
      $("#table{{event.name}}").hide();
      {% endfor %}

      if (value ≠ "−") {
        $("#guestsForm").show();
        document.getElementById("guestsFormTitle").innerHTML =
          value + " Event Subscription";
        $("#table" + value).show();
        $("#deleteEvent").show();
      }

      {% for event in events %}
      if (value ≡ "{{event.name}}") {
        document.getElementById("eventCapacity").innerHTML
          = 'Event Capacity: ' + "{{event.capacity}}";
        document.getElementById("eventGuests").innerHTML
```

```
            = 'Suscribed Guests: ' + "{{event.guests}}";
        }
        {% endfor %}

        return;
      }
    </script>

    </body>
</html>
{% endautoescape %}
```



```
from Model.Model import *
from Model.Model import Event

import webapp2
import urllib


class EventsCreation(webapp2.RequestHandler):
    def post(self):
        # Get all the events to check if the event already exists
        name = self.request.get('eventName')
        capacity = int(self.request.get('eventCapacity'))

        # Upload a new event if this name and capacity are valid
        if name:
            event = Event(parent=events_key())
            event.name = name
            event.capacity = str(capacity)
            event.guests = str(0)
            event.put()

        query_params = {'event_name': name}
        self.redirect('/?' + urllib.urlencode(query_params))
```

Oct 19, 15 15:37 **EventRemoval.py** Page 1/1

```
from google.appengine.ext import ndb
from Model.Model import *

import webapp2
import json
import logging


class EventRemoval(webapp2.RequestHandler):
    @ndb.transactional(xg=True)
    def delete_event(self, event_name):
        events_query = Event.query(ancestor=events_key())
        events = events_query.fetch()

        found = False
        for event in events:
            if event.name ≡ event_name:
                logging.warning("DELETING EVENT " + event_name)
                event.key.delete()
                found = True

        if ¬ found:
            return False

        # Delete all the guests related with this event
        guests_query = Guest.query(ancestor=event_guests_key(event_name))
        guests = guests_query.fetch()

        logging.warning("DELETING GUESTS OF EVENT" + event_name)
        for guest in guests:
            guest.key.delete()

        return True

    def post(self):
        actual_event = self.request.get('event')
        logging.debug("Event to remove: " + actual_event)

        if self.delete_event(actual_event):
            self.response.headers['Content-Type'] = "application/json"
            self.response.out.write(json.dumps({'deleted' : 'true'}))
        else:
            self.response.headers['Content-Type'] = "application/json"
            self.response.out.write(json.dumps({'deleted' : 'false'}))
```

Oct 19, 15 11:32 **AddGuest.py** Page 1/1

```
from google.appengine.ext import ndb
from Model.Model import *

import webapp2
import json
import logging

class AddGuest(webapp2.RequestHandler):
    @ndb.transactional(xg=True)
    def increment_guest_count(self, event_name):
        # Get the event
        events_query = Event.query(Event.name ≡ event_name,
                                   ancestor=events_key())
        event = events_query.get()

        logging.debug(
            "Event: " + event_name +
            " - Capacity: " + event.capacity)
        logging.debug(
            "Event: " + event_name +
            " - Guests: " + event.guests)
        if int(event.capacity) > int(event.guests):
            logging.debug(
                "Event " + event_name +
                ": There is capacity")
            amount_guests = int(event.guests) + 1
            event.guests = str(amount_guests)
            event.put()
            return True
        else:
            logging.debug(
                "Event " + event_name +
                ": There ISN'T capacity")
            return False

    def guest_exists(self, guest_email, event):
        guest_query = Guest.query(Guest.email ≡ guest_email,
                                  ancestor=event_guests_key(event))
        guest = guest_query.get()
        if guest:
            logging.warning(
                "Guest with email " + guest_email
                + " already exists. Dropping request")
            return True
        return False

    def post(self):
        event_name = self.request.get('actualEvent')
        check_duplicates = self.request.get('checkDuplicates')

        if check_duplicates ≡ "true" ∧\
            self.guest_exists(self.request.get('guestEmail'), event_name):
            self.response.headers['Content-Type'] = "application/json"
            self.response.out.write(json.dumps({'add': 'duplicated'}))
            return

        if ¬ self.increment_guest_count(event_name):
            self.response.headers['Content-Type'] = "text/plain"
            self.response.out.write(json.dumps({'add': 'false'}))
            return
        else:
            # Add the guest
            guest = Guest(parent=event_guests_key(event_name))
            guest.name = self.request.get('guestName')
            guest.surname = self.request.get('guestSurname')
            guest.email = self.request.get('guestEmail')
            guest.company = self.request.get('guestCompany')
            guest.put()

            self.response.headers['Content-Type'] = "application/json"
            self.response.out.write(json.dumps({'add': 'true'}))

```

Oct 19, 15 11:11 **application.py** Page 1/1

```
import os
from Model.Model import *
from Handlers.AddGuest import AddGuest
from Handlers.EventsCreation import EventsCreation
from Handlers.EventRemoval import EventRemoval

import jinja2
import webapp2

JINJA_ENVIRONMENT = jinja2.Environment(
    loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
    extensions=['jinja2.ext.autoescape'],
    autoescape=True)


class MainPage(webapp2.RequestHandler):
    def get(self):
        events_query = Event.query(ancestor=events_key())
        events = events_query.fetch()

        guest_list = []
        for event in events:
            guest_query = Guest.query(ancestor=event_guests_key(event.name))
            guests_by_event = guest_query.fetch()
            guest_list.append((event.name, guests_by_event))

        template_values = {
            'events': events,
            'guests_list': guest_list
        }

        template = JINJA_ENVIRONMENT.get_template('index.html')
        self.response.write(template.render(template_values))


app = webapp2.WSGIApplication([
    ('/', MainPage),
    ('/updater', MainPage),
    ('/event_creation', EventsCreation),
    ('/event_removal', EventRemoval),
    ('/add_guest', AddGuest),
], debug=True)
```

Oct 19, 15 15:43 **Table of Content** Page 1/1

Table of Contents