

```

1  from flask import Flask, session, redirect, url_for, escape, request, render_template, Blueprint, abort
2  import user
3  import sys
4  import bcrypt
5  import random
6  import debug components
7  import datetime
8  from models import Observation, User, Comment, UnitProcess, Reference, \
9      ComponentGroup, DataPointGroup, DataPoint, DataPointValue, \
10     DataPointGroupNote
11 from models import db
12
13 def construct_blueprint(settings):
14     blueprint = Blueprint(
15         'observations',
16         name,
17         template_folder='templates')
18
19     # This will return the template for the observations page. It has
20     # been abstracted for re-use.
21     def normal_observations_page():
22         user = db.session.query(User) \
23             .filter_by(id=session['user_id']).one()
24         observations = db.session.query(Observation).all()
25         unit_processes = db.session.query(UnitProcess).all()
26         return render_template(
27             'observations.html',
28             user = user,
29             observations = observations,
30             unit_processes = unit_processes,
31             datetime=datetime
32         )
33
34     @blueprint.route('/observations', methods=['GET', 'POST'])
35     def observations():
36         if 'user_id' not in session:
37             return redirect(url_for('logout.logout'))
38         user = db.session.query(User) \
39             .filter_by(id=session['user_id']).one()
40
41         if request.method == 'GET':
42             return normal_observations_page()
43         elif request.method == 'POST':
44             form_component_id = request.form.get('create_unit_process_id')
45             # form_component_id should be validated
46             # ...
47
48             if form_component_id is not None:
49                 try:
50                     form_component_id = int(form_component_id)
51                     # Get the form component with this ID, synthesize it,
52                     # and add it to an observation, then add it to the stack
53                     c = db.session.query(UnitProcess) \
54                         .filter_by(id=form_component_id).one()
55                     newobs = Observation(unit_process=c)
56                     db.session.add(newobs)
57                     db.session.commit()
58                 except:
59                     db.session.rollback()
60                 return normal_observations_page()
61             return 'post request not implemented', 500
62         else:
63             return 'Method not allowed', 405
64
65     @blueprint.route('/observations/<int:obs_id>', methods=['GET'])
66     def get_obs_id(obs_id):
67         if 'user_id' not in session:
68             return redirect(url_for('logout.logout'))
69         user = db.session.query(User) \
70             .filter_by(id=session['user_id']).one()
71         obs = db.session.query(Observation) \
72             .filter_by(id=obs_id).one()
73         return render_template('observation.html', user=user,
74                                observation=obs)
75
76     @blueprint.route('/observations/<int:obs_id>/edit', methods=['POST'])

```

```

77 def edit_obs_id(obs_id):
78     if 'user_id' not in session:
79         return redirect(url_for('logout.logout'))
80     user = db.session.query(User) \
81         .filter_by(id=session['user_id']).one()
82     observation = db.session.query(Observation) \
83         .filter_by(id=obs_id).one()
84     dp_errors = {}
85     return_anchor=None
86     for key in request.form:
87         value = request.form[key]
88         if value == '':
89             value = None
90         if key.startswith('DataPoint_'):
91             dp_id = int(key.split('_')[1])
92             dpv = None
93
94             # Attempt to locate a record for the DataPointValue
95             try:
96                 dpv = db.session.query(DataPointValue) \
97                     .filter_by(observation=observation,
98                               data_point_id=dp_id).one()
99
100            except:
101                print >> sys.stderr, "Could not find DPV in database"
102                pass
103
104            if dpv and dpv.value and value is None:
105                # Remove the DataPointValue
106                try:
107                    db.session.delete(dpv)
108                    db.session.commit()
109                    continue
110                except:
111                    print >> sys.stder, "Could not remove DataPointValue"
112                    db.session.rollback()
113
114            if value is not None:
115                try:
116                    dpv = DataPointValue(observation=observation,
117                                        data_point_id=dp_id,
118                                        value=value)
119
120                    db.session.add(dpv)
121                    db.session.commit()
122                except AssertionError as e:
123                    dp_errors[dp_id] = e.args[0]
124                    db.session.rollback()
125                except Exception as e:
126                    print >> sys.stderr, ("Could not write DataPointValue to database", e)
127
128            if key.startswith('DataPointGroupNote_'):
129                dpg_id = int(key.split('_')[1])
130                try:
131                    dpgn = db.session.query(DataPointGroupNote) \
132                        .filter_by(observation=observation,
133                                  data_point_group_id=dpg_id) \
134                        .one()
135                except:
136                    dpgn = DataPointGroupNote(observation=observation,
137                                              data_point_group_id=dpg_id,
138                                              text=None)
139
140                db.session.add(dpgn)
141
142                try:
143                    if dpgn.text and dpgn.text != '' and value is None:
144                        db.session.delete(dpgn)
145                    else:
146                        dpgn.text = value
147                        db.session.commit()
148                except:
149                    db.session.rollback()
150                    print >> sys.stderr, "Could not add DataPointGroupNote to observation."
151
152            if key == 'return_anchor' and value and value != '':
153                value = value.strip('#')
154                return_anchor=value
155
156            if dp_errors:
157                return render_template('observation.html', user=user,
158                                     observation=observation,
159                                     dp_errors=dp_errors)

```

```
153         return redirect(url_for('observations.get_obs_id',
154                                 obs_id=obs_id,
155                                 _anchor=return_anchor))
156
157
158 @blueprint.route('/observations/<int:obs_id>/comment', methods=['POST'])
159 def post_comment(obs_id):
160     if 'user_id' not in session:
161         return redirect(url_for('logout.logout'))
162     user = db.session.query(User) \
163           .filter_by(id=session['user_id']).one()
164     try:
165         comment_text = request.form.get('comment_text')
166         obs = db.session.query(Observation) \
167               .filter_by(id=obs_id).one()
168         # Validate the comment text
169         # ...
170         new_comment = Comment(user=user, text=comment_text, \
171                               observation=obs)
172         db.session.add(new_comment)
173         db.session.commit()
174     except:
175         pass
176     return redirect(url_for('observations.get_obs_id',
177                             obs_id=obs_id,
178                             _anchor='comments'))
179
180 return blueprint
181
```