Three Django code snippets from Ron Newman:

1. Simple Form Handling + related calls: contactus()
2. More Django-related Stuff: shareMap()
3. Recursive Connected Graph Traversal: climbTree()

Simple Form Handling

views.py ----------

```
@csrf_protect
def contactus(request):
  try:
    context = {'ideatrees_email': settings.EMAIL_OF_IDEATREE}
    if request.method == 'GET':
      form = ContactUsForm(label_suffix='')
      context["form"] = form
      return render(request,'ideatree/contactus.html', context)
    if request.method == 'POST':
      userId = request.session["_auth_user_id"] if request.session.keys() else None
      form = ContactUsForm(request.POST, label_suffix='')
      context["form"] = form
      if not form.is_valid():
        return render(request,'ideatree/contactus.html', context )
      else:
        email = request.POST['email']
        userMsg = request.POST['message']
        if userId:
          ContactUs.objects.create(user_id=userId, email=email, message=userMsg)
        fromAddr = settings.CONTACTUS_EMAIL_FROM_ADDR
        subject="IDEATREE SUPPORT INQUIRY"
        msg="Message from:" + email + "      " + userMsg
        try:
          send_mail(subject, msg, fromAddr, [email], fail_silently=False,)
          return HttpResponse(settings.CONTACTUS_SUCCESS_MESSAGE)
        except Exception as err:
          raise Exception("Error while sending your message: " + str(err))
  except Exception as err:
```

forms.py ----------

```
class ContactUsForm(ModelForm):
  #captcha = ReCaptchaField()
  class Meta:
    model = ContactUs
    fields = ['email','message']
    widgets = {'message': Textarea(attrs={'placeholder':'Are you using Chrome, Firefox, Safari,
other?\n\nOn Windows, Mac, Linux, other?'}) }
```

urls.py (excerpt) ----------

```
app_name = 'ideatree'
urlpatterns = [
  re_path(r'^map/contactus/$', views.contactus, name='contactus'),
]
```

More Django-related Stuff

prefetch_selected, lambda, ORM queries, etc.

Original SQL Query replaced by ORM:

```
#   friendQuery = "SELECT DISTINCT ideatree_friend.id, auth_user.username, first_name,
last_name, auth_user.email, ideatree_map_desc.mapname, ideatree_map_desc.id as ma
#     FROM ideatree_friend\
#     LEFT JOIN auth_user ON auth_user.id=ideatree_friend.friend_id AND
ideatree_friend.status !="'+ settings.FRIEND_DELETED + "' AND ideatree_friend.initiator_id =
#     LEFT JOIN ideatree_mapmember ON
ideatree_mapmember.member_id=ideatree_friend.friend_id AND
ideatree_mapmember.status !="'+ settings.MAPMEMBERSHIP_DELETED + "' \
#     LEFT JOIN ideatree_map_desc ON
ideatree_map_desc.id=ideatree_mapmember.ofmap_id AND
ideatree_map_desc.owner_id="+ thisUser +" \
#     WHERE auth_user.id !="+ thisUser + " AND mapname IS NOT NULL"
```

views.py ----------

```
@user_passes_test(not_guest_user)
@login_required()
@ensure_csrf_cookie  # Because the form/render combination is not used.
@transaction.atomic()
def shareMap(request):
  try:
    thisUserId = int(request.session["_auth_user_id"])

    # FIXME: do this sort with django-tables2 API
    sortby = request.POST['?sort'] if '?sort' in request.POST else None

    sharedWith = []
    myfriends = Friend.objects.filter(initiator_id=thisUserId,
status=settings.FRIEND_ACCEPTED)
    for fr in myfriends:
      mm = Mapmember.objects.filter(member=fr.friend_id, ofmap__owner_id=thisUserId,
status=settings.MAPMEMBERSHIP_ACTIVE).select_related('ofmap','member')
    for m in mm: # member in Mapmembers
        friend = {}
```

```
        friend["mapname"]=m.ofmap.mapname
        friend["friend_id"]=m.member.id
        friend["username"]=m.member.username
        friend["first_name"]=m.member.first_name
        friend["last_name"]=m.member.last_name
        sharedWith.append(friend)
    if sortby:
      sharedWith = sorted(sharedWith, key=lambda k: k['username'])
    table = FriendTable(sharedWith)
    RequestConfig(request).configure(table)
    context = {'table':table, 'title':settings.WHAT_A_GRAPH_IS_CALLED.capitalize()+'s You\'ve
Shared', 'navbuttontext':"Share Current "+settings.WHAT_A_GRAPH_IS_CALLED.
    return render(request, "ideatree/searchUserResults.html", context )

  except Warning as err:
    context = {'error':str(err)}
    return render(request, "showWarning.html", context, content_type="application/html")
  except FieldError as err:
    return HttpResponse(str(err), status=406)
  except Exception as err:
    return HttpResponse(str(err), status=406)

shareMap.alters_data = True
```

Recursive Connected Graph Traversal
This is what you invoke when clicking ‘View->Outline” within IdeaTreeLive.

views.py ----------

```
#**
# Depth-first recursive traversal of a multi-child tree (a non-directed multigraph).
# Outputs an html representation of the graph, in outline form.
# @author     Ron Newman <ron.newman@gmail.com>
# @copyright Ron Newman 2016
# @version   0.9
#/

def climbTree(mapId, edge, lastWroteNewline, indentlevel):
  try:
  if edge:
    # Mark our trail so that we don't retrieve this edge again
    #print("marking edge:" + edge.label + " id:" + str(edge.id))
    Edge.objects.filter(pk=edge.id, ofmap=mapId).update(pheremone=True)
```

```
    indentlevel += 1
    if edge.pheremone:
      #print("disregarding edge:" + edge.label + " id:" + str(edge.id))
      return(True)
    originId = edge.origin_id
    targetId = edge.target_id
    edgelabel = makeLabel(edge, settings.OUTLINE_EDGE_LABEL_LENGTH)
    if not edgelabel:
      edgelabel = "&rarr;"
    else:
      edgelabel = "&horbar;<span class='italicText'>"+ edgelabel +"</span>&rarr;"
    # TODO: Look into double direction tree traversal for possible greater efficiency.
    # See graphviz.org archives, search for 'dijkstra'
    origin = Node.objects.get(pk=originId, ofmap=mapId)
    originLabel = makeLabel(origin, settings.OUTLINE_NODE_LABEL_LENGTH)
    if origin.url:
      originLabel = "<a href='"+ str(origin.url) + "'>"+ originLabel + "</a>"
      # show the origin node
      out = ''
      if lastWroteNewline:
          for i in range(1,indentlevel,1):
            # matching indent of the corresponding cell in the row above,
            # one cell for content of the row above, one for edge
            out += settings.OUTLINE_CELL_DIV_START   + “ </div>” +
settings.OUTLINE_EDGE_LABEL_DIV_START + “ </div>"
            lastWroteNewline = False
            out += settings.OUTLINE_CELL_DIV_START + originLabel + "</div>"

    # show the edge
    out += settings.OUTLINE_EDGE_LABEL_DIV_START> + edgelabel + "</div> "

    # show the target node
    # See if this target node has children by looking for edges emanating from it.
    edgesFromChild = Edge.objects.filter(origin=targetId, pheremone=False,
status=settings.EDGE_ACTIVE, ofmap=mapId)

    # Notice that target becomes origin on the recursion
    if not edgesFromChild.count():  # no grandchildren to continue searching, just get the child
      target = Node.objects.get(pk=targetId, ofmap=mapId)
      targetLabel = makeLabel(target, settings.OUTLINE_NODE_LABEL_LENGTH)
      out += settings.OUTLINE_CELL_DIV_START + targetLabel + "</div>"
      out += "<br clear='both'/>"
      lastWroteNewline = True
    else:  # has children
      for edge in edgesFromChild:
        out += climbTree(mapId, edge, lastWroteNewline, indentlevel)    # start the recursion
        # unwinding recursive call stack
        lastWroteNewline = True
    return(out)
```

```
  except Exception as err:
    return HttpResponse(str(err), status=406)
```

(climbTree() calling routings are omitted, they’re trivial)