**from** pytz **import** all\_timezones\_set

*# -\*- coding: utf-8 -\*-
# this file is released under public domain and you can use without limitations

# -------------------------------------------------------------------------
# This is a sample controller
# - index is the default action of any application
# - user is required for authentication and authorization
# - download is for downloading files uploaded in the db (does streaming)
# -------------------------------------------------------------------------***def** index():
 *"""
 example action using the internationalization operator T and flash
 rendered by views/default/index.html or views/generic.html

 if you need a simple wiki simply replace the two lines below with:
 return auth.wiki()
 """* **return** dict(message=T(**'Welcome to web2py!'**))

**def** user():
 *"""
 exposes:
 http://..../[app]/default/user/login
 http://..../[app]/default/user/logout
 http://..../[app]/default/user/register
 http://..../[app]/default/user/profile
 http://..../[app]/default/user/retrieve\_password
 http://..../[app]/default/user/change\_password
 http://..../[app]/default/user/bulk\_register
 use @auth.requires\_login()
 @auth.requires\_membership('group name')
 @auth.requires\_permission('read','table name',record\_id)
 to decorate functions that need access control
 also notice there is http://..../[app]/appadmin/manage/auth to allow administrator to manage users
 """* **return** dict(form=auth())

@cache.action()
**def** download():
 *"""
 allows downloading of uploaded files
 http://..../[app]/default/download/[filename]
 """* **return** response.download(request, db)

**def** call():
 *"""
 exposes services. for example:
 http://..../[app]/default/call/jsonrpc
 decorate with @services.jsonrpc the functions to expose
 supports xml, json, xmlrpc, jsonrpc, amfrpc, rss, csv
 """* **return** service()

**def** define\_timezone():
 *"""Call ajax to determine user timezone"""* tz\_name = request.vars.name

 **if** tz\_name **in** all\_timezones\_set:
 session.user\_timezone = tz\_name

@auth.requires\_login()
**def** manage\_events():
 **if 'new' in** request.args:
 redirect(URL(**'new\_event'**))
 **elif 'edit' in** request.args:
 redirect(URL(**'edit\_event'**, args=[request.args(2)]))
 form = SQLFORM.grid(db.events, searchable=True, editable=True, deletable=True, details=False,
 create=True, paginate=20, maxtextlength=60, fields=[db.events.event\_time,
 db.events.e\_type,
 db.events.e\_level,
 db.events.b\_scale,
 db.events.systolic,
 db.events.diastolic,
 db.events.pulse,
 db.events.medicine\_name,
 db.events.lbs,
 db.events.duration\_time,
 db.events.note],
 orderby=~db.events.event\_time)
 **return** dict(form=form)

@auth.requires\_login()
**def** new\_event():
 db.events.systolic.show\_if = (db.events.e\_type == 1)
 db.events.diastolic.show\_if = (db.events.e\_type == 1)
 db.events.pulse.show\_if = (db.events.e\_type == 1)
 db.events.medicine\_name.show\_if = (db.events.e\_type == 3)
 db.events.e\_level.show\_if = (db.events.e\_type.belongs(4, 2, 5, 6))
 db.events.duration\_time.show\_if = (db.events.e\_type.belongs(2, 4, 5, 6, 7, 10))
 db.events.b\_scale.show\_if = (db.events.e\_type == 8)
 db.events.lbs.show\_if = (db.events.e\_type == 9)
 form = SQLFORM(db.events, fields=[**"event\_time"**,
 **"e\_type"**,
 **"e\_level"**,
 **"b\_scale"**,
 **"systolic"**,
 **"diastolic"**,
 **"pulse"**,
 **"medicine\_name"**,
 **"lbs"**,
 **"duration\_time"**,
 **"note"**])
 **if** form.process().accepted:
 response.flash = **'Thanks for filling out the form'** redirect(URL(**'manage\_events'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** edit\_event():
 this\_event = db.event\_types(db.events.id==request.args(0,cast=int))
 db.events.systolic.show\_if = (db.events.e\_type == 1)
 db.events.diastolic.show\_if = (db.events.e\_type == 1)
 db.events.pulse.show\_if = (db.events.e\_type == 1)
 db.events.medicine\_name.show\_if = (db.events.e\_type == 3)
 db.events.e\_level.show\_if = (db.events.e\_type.belongs(4, 2, 5, 6))
 db.events.duration\_time.show\_if = (db.events.e\_type.belongs(2, 4, 5, 6, 7, 10))
 db.events.b\_scale.show\_if = (db.events.e\_type == 8)
 db.events.lbs.show\_if = (db.events.e\_type == 9)
 form=SQLFORM(db.events, this\_event, fields=[**"event\_time"**,
 **"e\_type"**,
 **"e\_level"**,
 **"b\_scale"**,
 **"systolic"**,
 **"diastolic"**,
 **"pulse"**,
 **"medicine\_name"**,
 **"lbs"**,
 **"duration\_time"**,
 **"note"**])
 **if** form.process().accepted:
 response.flash=**'Thanks for editing the form'** redirect(URL(**'manage\_events'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** manage\_medicines():
 **if 'new' in** request.args:
 redirect(URL(**'new\_medicine'**))
 **elif 'edit' in** request.args:
 redirect(URL(**'edit\_medicine'**, args=[request.args(2)]))
 form = SQLFORM.grid(db.medicines, searchable=True, editable=True, deletable=True, details=False,
 create=True, paginate=20, maxtextlength=60, fields=[db.medicines.medicine\_name,
 db.medicines.dosage],
 orderby=db.medicines.medicine\_name)
 **return** dict(form=form)

@auth.requires\_login()
**def** new\_medicine():
 form = SQLFORM(db.medicines, fields=[**"medicine\_name"**,
 **"dosage"**])
 **if** form.process().accepted:
 response.flash = **'Thanks for filling out the form'** redirect(URL(**'manage\_medicines'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** edit\_medicine():
 this\_medicine = db.medicines(db.medicines.id==request.args(0,cast=int))
 form=SQLFORM(db.medicines, this\_medicine, fields=[**"medicine\_name"**,
 **"dosage"**])
 **if** form.process().accepted:
 response.flash=**'Thanks for editing the form'** redirect(URL(**'manage\_medicines'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** manage\_bristol\_scales():
 **if 'new' in** request.args:
 redirect(URL(**'new\_bristol\_scale'**))
 **elif 'edit' in** request.args:
 redirect(URL(**'edit\_bristol\_scale'**, args=[request.args(2)]))
 form = SQLFORM.grid(db.bristol\_scales, searchable=True, editable=True, deletable=True, details=False,
 create=True, paginate=20, maxtextlength=60, fields=[db.bristol\_scales.b\_scale],
 orderby=db.bristol\_scales.b\_scale)
 **return** dict(form=form)

@auth.requires\_login()
**def** new\_bristol\_scale():
 form = SQLFORM(db.bristol\_scales, fields=[**"b\_scale"**])
 **if** form.process().accepted:
 response.flash = **'Thanks for filling out the form'** redirect(URL(**'manage\_bristol\_scales'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** edit\_bristol\_scale():
 this\_bristol\_scale = db.bristol\_scales(db.bristol\_scales.id==request.args(0,cast=int))
 form=SQLFORM(db.bristol\_scales, this\_bristol\_scale, fields=[**"b\_scale"**])
 **if** form.process().accepted:
 response.flash=**'Thanks for editing the form'** redirect(URL(**'manage\_bristol\_scales'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** manage\_event\_types():
 **if 'new' in** request.args:
 redirect(URL(**'new\_event\_type'**))
 **elif 'edit' in** request.args:
 redirect(URL(**'edit\_event\_type'**, args=[request.args(2)]))
 form = SQLFORM.grid(db.event\_types, searchable=True, editable=True, deletable=True, details=False,
 create=True, paginate=20, maxtextlength=60, fields=[db.event\_types.e\_type],
 orderby=db.event\_types.e\_type)
 **return** dict(form=form)

@auth.requires\_login()
**def** new\_event\_type():
 form = SQLFORM(db.event\_types, fields=[**"e\_type"**])
 **if** form.process().accepted:
 response.flash = **'Thanks for filling out the form'** redirect(URL(**'manage\_event\_types'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** edit\_event\_type():
 this\_event\_type = db.event\_types(db.event\_types.id==request.args(0,cast=int))
 form=SQLFORM(db.event\_types, this\_event\_type, fields=[**"e\_type"**])
 **if** form.process().accepted:
 response.flash=**'Thanks for editing the form'** redirect(URL(**'manage\_event\_types'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** manage\_event\_levels():
 **if 'new' in** request.args:
 redirect(URL(**'new\_event\_level'**))
 **elif 'edit' in** request.args:
 redirect(URL(**'edit\_event\_level'**, args=[request.args(2)]))
 form = SQLFORM.grid(db.event\_levels, searchable=True, editable=True, deletable=True, details=False,
 create=True, paginate=20, maxtextlength=60, fields=[db.event\_levels.e\_level],
 orderby=db.event\_levels.e\_level)
 **return** dict(form=form)

@auth.requires\_login()
**def** new\_event\_level():
 form = SQLFORM(db.event\_levels, fields=[**"e\_level"**])
 **if** form.process().accepted:
 response.flash = **'Thanks for filling out the form'** redirect(URL(**'manage\_event\_levels'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** edit\_event\_level():
 this\_event\_level = db.event\_levels(db.event\_levels.id==request.args(0,cast=int))
 form=SQLFORM(db.event\_levels, this\_event\_level, fields=[**"e\_level"**])
 **if** form.process().accepted:
 response.flash=**'Thanks for editing the form'** redirect(URL(**'manage\_event\_levels'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** manage\_durations():
 **if 'new' in** request.args:
 redirect(URL(**'new\_duration'**))
 **elif 'edit' in** request.args:
 redirect(URL(**'edit\_duration'**, args=[request.args(2)]))
 form = SQLFORM.grid(db.durations, searchable=True, editable=True, deletable=True, details=False,
 create=True, paginate=20, maxtextlength=60, fields=[db.durations.duration\_time],
 orderby=db.durations.duration\_time)
 **return** dict(form=form)

@auth.requires\_login()
**def** new\_duration():
 form = SQLFORM(db.durations, fields=[**"duration\_time"**])
 **if** form.process().accepted:
 response.flash = **'Thanks for filling out the form'** redirect(URL(**'manage\_durations'**))
 **return** dict(form=form)

@auth.requires\_login()
**def** edit\_duration():
 this\_duration = db.durations(db.durations.id==request.args(0,cast=int))
 form=SQLFORM(db.durations, this\_duration, fields=[**"duration\_time"**])
 **if** form.process().accepted:
 response.flash=**'Thanks for editing the form'** redirect(URL(**'manage\_durations'**))
 **return** dict(form=form)