Django FSQ

[https://github.com/davidgillies/django-fsq]

Django-fsq is a packaged app version of the models for the Forms System MySQL Database. You need to have Django installed and add it to your project.

Features

It provides models to make it easy to connect up to the form system and use Django's Object Relational Mapper (ORM) to work with the database. It can be set up to use a local Sqlite database or another existing database.

It provides a router that allows connections to non-default databases.

It connects the models to Django's admin system so they can be managed there. There is export functionality for each table.

Django-fsq provides Users, Groups, Progress, Questionnaires, Results and Roles models, that match up to the Forms System MySQL Database. The classes:

```
class Users(models.Model):
    user_id = models.CharField(primary_key=True, max_length=50)
    password_hash = models.TextField(blank=True)
    salt = models.CharField(max_length=50, blank=True)

    def __unicode__(self):
        return self.user_id

    class Meta:
        db_table = 'users'
        verbose_name = 'User'
        verbose_name_plural = 'Users'


class Groups(models.Model):
    user_id = models.CharField(primary_key=True, max_length=50)
    study_name = models.CharField(max_length=50)
    user_group = models.CharField(max_length=50, blank=True)

    def __unicode__(self):
        return "%s, %s" % (self.user_group, self.user_id)

    class Meta:
        db_table = 'groups'
        verbose_name = 'Group'
        verbose_name_plural = 'Groups'


class Progress(models.Model):
    user = models.ForeignKey(Users)
```

```python
    questionnaire_id = models.CharField(max_length=50, blank=True)
    started = models.IntegerField(blank=True, null=True)
    finished = models.IntegerField(blank=True, null=True)
    progress_id = models.AutoField(primary_key=True)

    def __unicode__(self):
        return self.user.user_id

    class Meta:
        db_table = 'progress'
        verbose_name = 'User Progress'
        verbose_name_plural = 'User Progress'


class Questionnaires(models.Model):
    id = models.AutoField(primary_key=True)
    role = models.CharField(max_length=50)
    study_name = models.CharField(max_length=50)
    user_group = models.CharField(max_length=50)
    questionnaire_id = models.CharField(max_length=50)
    questionnaire_name = models.CharField(max_length=50)
    save_as = models.CharField(max_length=45)

    def __unicode__(self):
        return "%s, %s" % (self.questionnaire_id, self.study_name)

    class Meta:
        db_table = 'questionnaires'
        verbose_name = 'Questionnaire'
        verbose_name_plural = 'Questionnaires'


class Results(models.Model):
    id = models.AutoField(primary_key=True)  # AutoField?
    user = models.ForeignKey(Users)
    questionnaire_id = models.CharField(max_length=45)
    var_id = models.CharField(max_length=45, blank=True)
    var_name = models.CharField(max_length=100)
    var_value = models.CharField(max_length=500, blank=True)

    def __unicode__(self):
        return self.user.user_id

    class Meta:
        db_table = 'results'
        verbose_name = 'User Result'
        verbose_name_plural = 'User Results'


class Roles(models.Model):
    user_id = models.CharField(primary_key=True, max_length=50)
    role = models.CharField(max_length=50)

    def __unicode__(self):
        return "%s, %s" % (self.user_id)

    class Meta:
        db_table = 'roles'
        verbose_name = 'Role'
        verbose_name_plural = 'Roles'
```

Installation

Django-fsq is a package which works with Django. Install with pip

```
pip install git+git://github.com/davidgillies/django-fsq
```

This will install it directly into your site-packages directory. If you are using virtualenv, this will be inside the Lib folder at your virtualenv's home directory. Alternatively if you can install it like this

```
pip install -e git+git://github.com/davidgillies/django-fsq#egg=django-fsq
```

This will create a source folder in your virtulenv home folder and clones the entire package to that folder. This requires git to be installed on your system. You can work with the code directly in this location and use git there as normal.

Config

There are 2 options for using the package.

Case 1 - Normal Django using default database

In your project settings, add to INSTALLED_APPS

```
'questionnaire',
'import_export',
```

then

```
python manage.py migrate
```

Case 2 - Set up with pre-existing MySQL database

For any pre-existing database you will have to set up the backend in your settings file in the normal way. For MySQL you can use pymysql

```
pip install pymsql
```

To use pymysql add in your manage.py immediately under `import sys`

```
    try:
        import pymysql
        pymysql.install_as_MySQLdb()
    except ImportError:
        pass
```

In your project settings, add to INSTALLED_APPS

```
'questionnaire',
'import_export',
```

Also in your project settings add your DB details, e.g. :

```
'db3': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'mrc_epid_fenland',
    'USER': 'david',
    'PASSWORD': '*****',
    'HOST': 'localhost',
    'PORT': '3306',
},
```

Also in the project settings add the setting

```
DATABASE_ROUTERS = ['questionnaire.routers.PlayRouter',]
```

The db name in your django settings has to be 'db3' in order for the routers in the questionnaire to connect up.

Uninstall

```
pip uninstall django-fsq
```

Usage

The questionnaire models are accessible via the Django Admin and data can be exported from there.

To work with the models in another app you can simply import them and use them in the usual way.

```
from questionnaire.models import Results, Users
```

```
results = Results.objects.all()
user = Users.objects.get(user_id='davidg')
```

Project Structure

This is a simple Django app. The root conatins setup.py etc. that define the installation process.

questionnaire/admin.py provides the admin screen set up and adds in the import/export tools.

questionnaire/models.py provides the project models described in the intro.

questionnaire/routers.py provides routers for a database with the key 'db3' that you can set up in your Django project settings.py DATABASES dictionary as suggested in the INstallation section.

Other files are just standard Django place holders.

Dependencies

django-import-export # provides import/export tools for the admin interface.