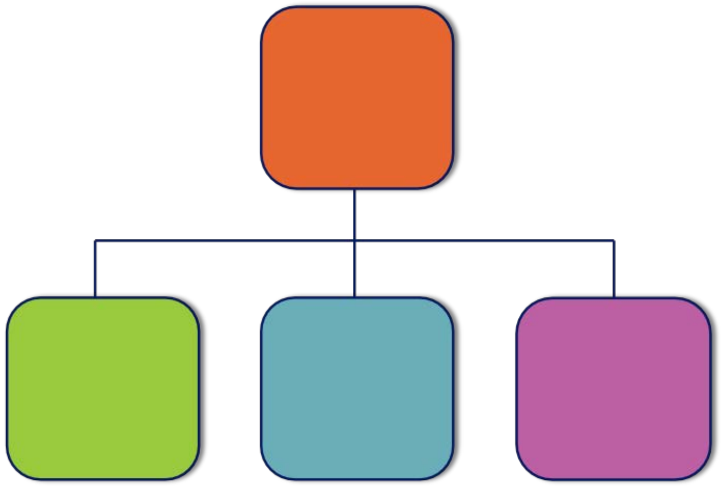
Models

Reindert-Jan Ekker
nl.linkedin.com/in/rjekker/
@rjekker

Overview



- **Models**
  - Writing Models
  - Field types
  - Saving and deleting data
  - Database queries
  - Relations
- **Generating the database**
- **The auto-generated admin UI**

Django Model Classes

- **Each model class maps to a database table**
  - subclasses `django.db.models.Model`
- **Each attribute of the model represents a database field**
  - should be an instance of the appropriate `Field` class.
  - Documentation for Fields: http://goo.gl/rgqWZu
- **Django uses the field class types for:**
  - The database column type (e.g. INTEGER, VARCHAR).
  - The default HTML widget to use when rendering a form field
- **Django generates a Model API**
  - For retrieving and storing data from Python code

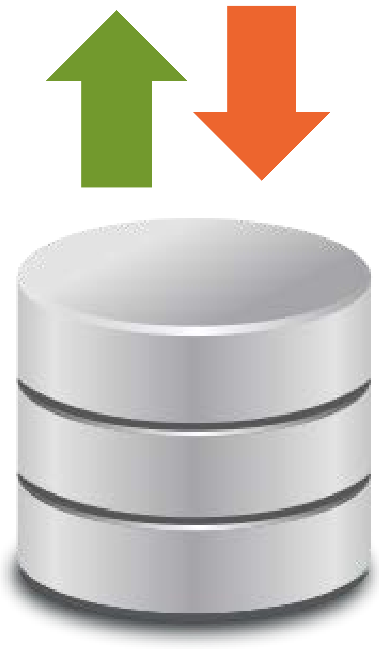
Manage.py Database Commands

- **`python manage.py sql appname`**
  - Prints CREATE TABLE SQL statements for the given app name
- **`python manage.py syncdb`**
  - Creates the database tables for all installed apps whose tables have not already been created
- **Syncdb does NOT do database migration!**
  - It will not alter tables
  - Migrations will be a part of Django 1.7
- **Changing a Model**
  - Drop the table; run syncdb
  - Use south (http://goo.gl/8n4qmA)

Admin

- **An auto-generated user interface to edit your data**
  - Need to register your models in your apps’ `admin.py`
  - `admin.site.register(MyModel)`
- **Very customizable**
  - For documentation, see: http://goo.gl/70YyPC
- **Implement __str__ for your Model classes**

Save and Delete

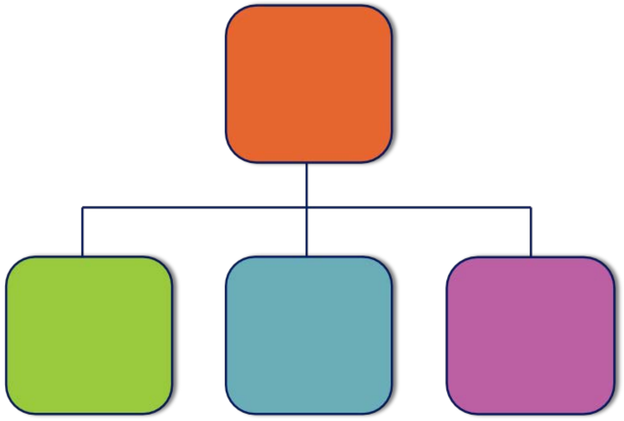


- Create a new instance with keyword arguments
  - `m=new Move(x=1,y=2,game=g)`
- save() on a new object:
  - `m.save()`
  - Sets the primary key field and does SQL INSERT
- save() on an existing object with changes: UPDATE
- delete() removes an object from the database
  - `g=Game.objects.get(pk=1); g.delete()`

The Model API

- **Model classes have a Manager instance called "objects"**
  - Is a class attribute: `Game.objects` not `g.objects`
- **get() returns a single instance**
  - `Game.objects.get(pk=1)`
- **all() returns all rows**
- **filter() returns matching objects**
  - `Game.objects.filter(status="A")`
- **exclude() returns objects that don't match**
- **Models documentation: http://goo.gl/RA0eT9**

One-to-Many Relations



- **Defined by a ForeignKey field**
  - On the "one" side of the relation
- **Many side gets a xxx_set attribute**
  - Where xxx is the name of the related model
  - This is a "related manager" object
  - Works just like "objects" manager
- **Set relation from Move m to Game g:**
  - m.game=g
  - or
  - g.move_set.add(m)
- **Django also offers OneToOne and ManyToMany fields (http://goo.gl/rgqWZu)**

Summary

- Writing Models
- Field types
- Generating the database
- Saving and deleting data
- Database queries
- Relations
- The auto-generated admin UI