Building a REST API Using Django & Django REST Framework

Kenny Yarboro

http://github.com/kennyncsu/drf-restaurant-api

Getting Started

- Quick Introductions
  - REST Architecture Overview
  - Django
  - Django REST Framework
- Necessary Setup
- Start Project

REST Architecture[1]

- Alternative Approach to Web Services
  - SOAP
- Typically HTTP-based
- Standard HTTP Methods
  - GET
  - PUT
  - POST
  - DELETE

REST Architecture[1]

- Client-Server Model
- Stateless
  - Links Determine State
- Cacheable
  - Define Cacheable or Not
- Layered System
  - Client Not Necessarily Communicating with End Server

REST Architecture[1]

- Uniform Interface
- Identification of Resources
- Hypermedia as the Engine of App. State
  - HATEOAS
- ‘Plug-and-Play’ Services

2

- High-level Python Web Framework
- Quickly Develop Applications
- Built-In Security
- Scalable
- Built-In Administrative Interface

- MVC vs MTV
- Model = Model
- View = Template
- Controller = View

- Extension Based on Django
- Toolkit to Easily Build Web APIs
- Built-In Web Browsable API
- Customizable
  - Basic Features
  - Enhanced Features

Necessary Setup

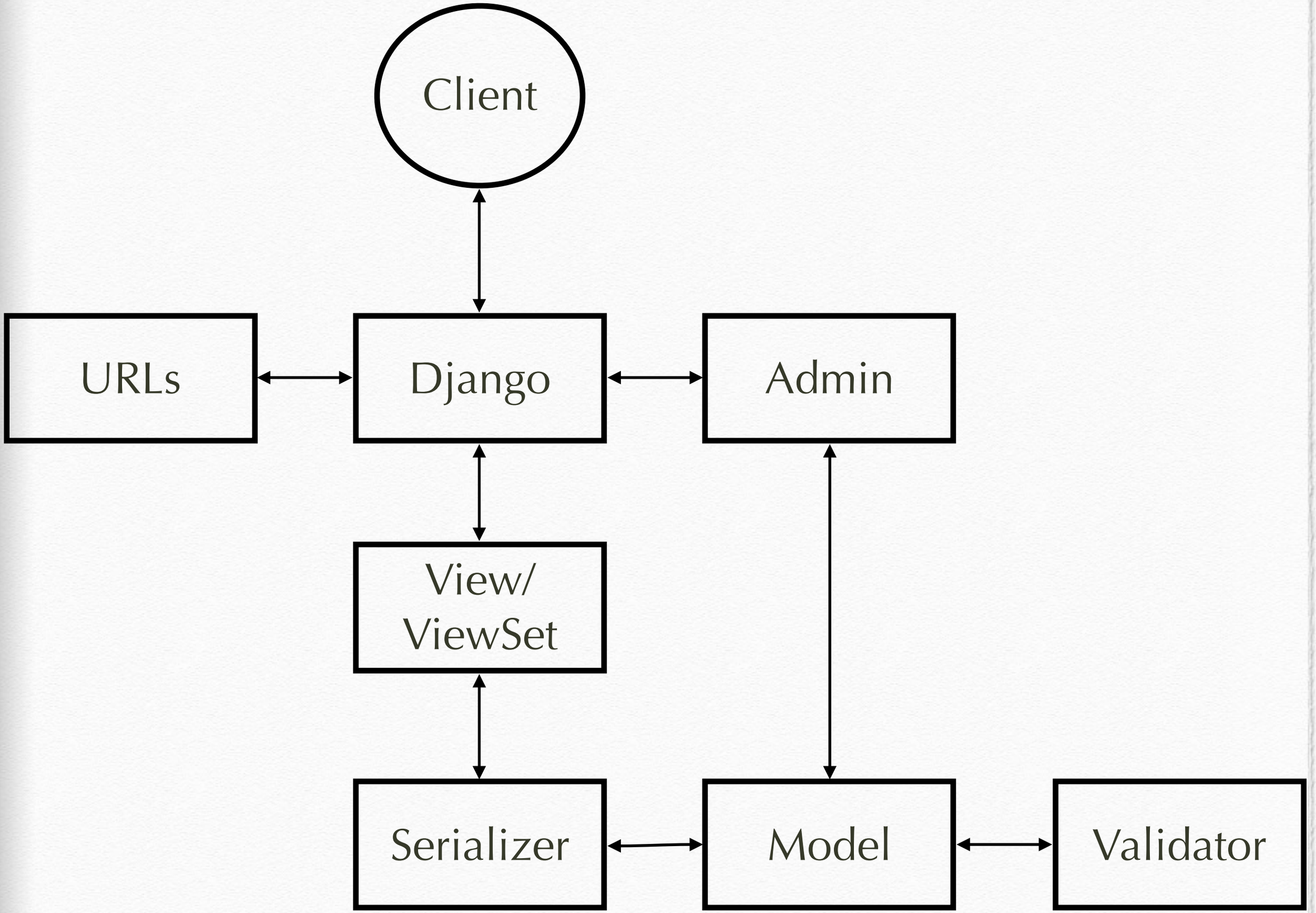
- Python 2 vs 3
- Python 2.x
  - Supported by More Libraries
  - 2.7 Pre-Loaded on Mac
- Python 3.x
  - Continually Building Support

Necessary Setup

- pip
  - https://bootstrap.pypa.io/get-pip.py
    - Windows: Add to System Path: C:\<path_to_python>\Scripts
  - python get-pip.py
- virtualenv
  - pip install virtualenv
  - cd <folder_to_use_for_project>
  - virtualenv venv
  - . venv/bin/activate
    - Windows: venv\Scripts\activate
  - deactivate

Necessary Setup

- Django
  - pip install django==1.7.4
- Django REST Framework (DRF)
  - pip install djangorestframework==3.0.4

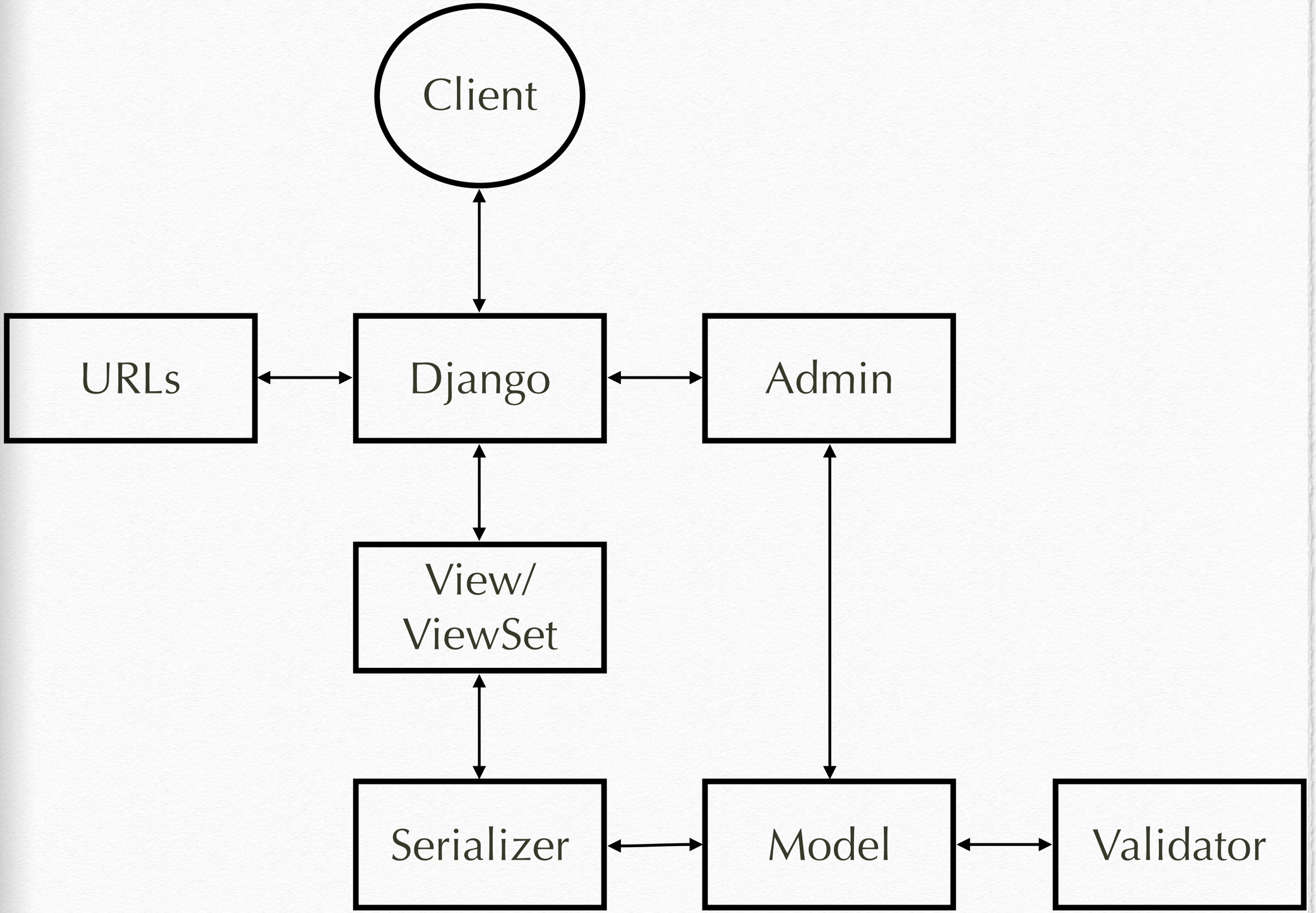

Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

Start Project

- Use Django to Create Project
  - django-admin.py/exe startproject tutorial .
- Several Files Created
  - manage.py - runs the project
  - tutorial Folder *(change to this directory)*
    - settings.py - configuration of project
    - urls.py - define addresses for resources
    - wsgi.py - default WSGI configuration for web server

Start Project

- Use Django to Create Application
  - django-admin.py/exe startapp restaurantapi
- Several Files Created
  - restaurantapi Folder
    - admin.py - set up Admin views
    - models.py - define data objects
    - views.py - callable functions or classes
      - Take Request, Return Response
    - migrations Folder - think version control for models


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

Start Project

- Create Initial Database
  - python manage.py migrate
- New File: db.sqlite3
- Create Superuser
  - python manage.py createsuperuser
- Start Project
  - python manage.py runserver --noreload
- http://localhost:8000

Restaurant Menu

- Restaurant Menu API Requirements
- Building the Application
- Create the Database
- Generating Data for the Application

Restaurant Menu API Requirements

- Types of Users
- Components of Menu
- Menu Actions

Types of Users

- Administrator/Manager
  - Manage the Application
  - Maintain Menu
    - Maintain Users & Permissions
- Chef
  - Maintain Menu
- Customers (Other)
  - View Menu

User & Group Models

- Default Models in Django
- Benefits
  - User Management
  - Password Encryption
  - Authentication
  - Group Permissions

Menu Model

- id - unique numeric value
- name - text
- description - text
- chef - ForeignKey: User
- available - boolean

Menu Item Model

- id - unique numeric value
- name - text
- description - text
- cost_to_make - Real number
- sale_price - Real number
- available - boolean
- menu - ForeignKey: Menu

Build Models
models.py

- Add Menu Model

```python
from django.contrib.auth.models import User
from django.db import models

# Create your models here.

class Menu(models.Model):

    name = models.CharField(max_length=32, unique=True)
    description = models.CharField(max_length=200)
    chef = models.ForeignKey(User)
    available = models.BooleanField(default=False)

    def __unicode__(self):
        return u'%s by %s %s' % (self.name, self.chef.first_name, self.chef.last_name)
```

Build Models
models.py

- Add Menu Item Model

```
17  class MenuItem(models.Model):
18
19      name = models.CharField(max_length=32, unique=True)
20      description = models.CharField(max_length=200)
21      cost_to_make = models.DecimalField(decimal_places=2, max_digits=5)
22      sale_price = models.DecimalField(decimal_places=2, max_digits=5)
23      available = models.BooleanField(default=False)
24      menu = models.ForeignKey(Menu)
25
26      def __unicode__(self):
27          return u'%s' % (self.name)
```

Modify Settings
settings.py

- Add Application to Installed Apps

```
32  INSTALLED_APPS = (
33      'django.contrib.admin',
34      'django.contrib.auth',
35      'django.contrib.contenttypes',
36      'django.contrib.sessions',
37      'django.contrib.messages',
38      'django.contrib.staticfiles',
39      'tutorial.restaurantapi',
40  )
```

Create the Database

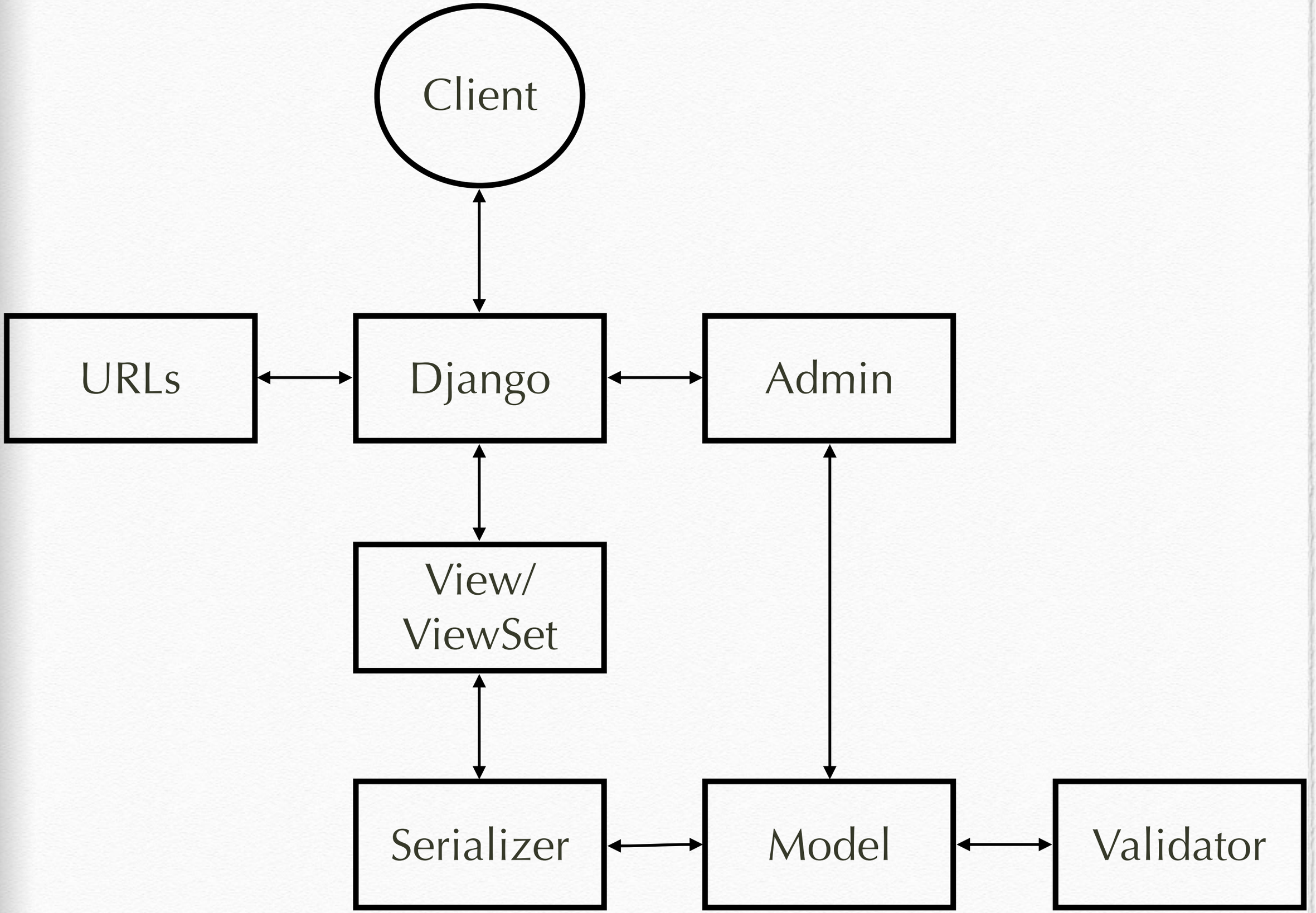
- Create Migrations for New Models
  - python manage.py makemigrations
- Commit New Migrations
  - python manage.py migrate
- Start Project
  - python manage.py runserver --noreload

Generating Data for the Application

- Create Menu
  - Get ID of New Menu
- Create Menu Item
  - Get ID of New Menu Item
- Query Database for Data

Generating Data for the Application

- Create More Menu Items
- Create Menus
- Query Database for Data


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

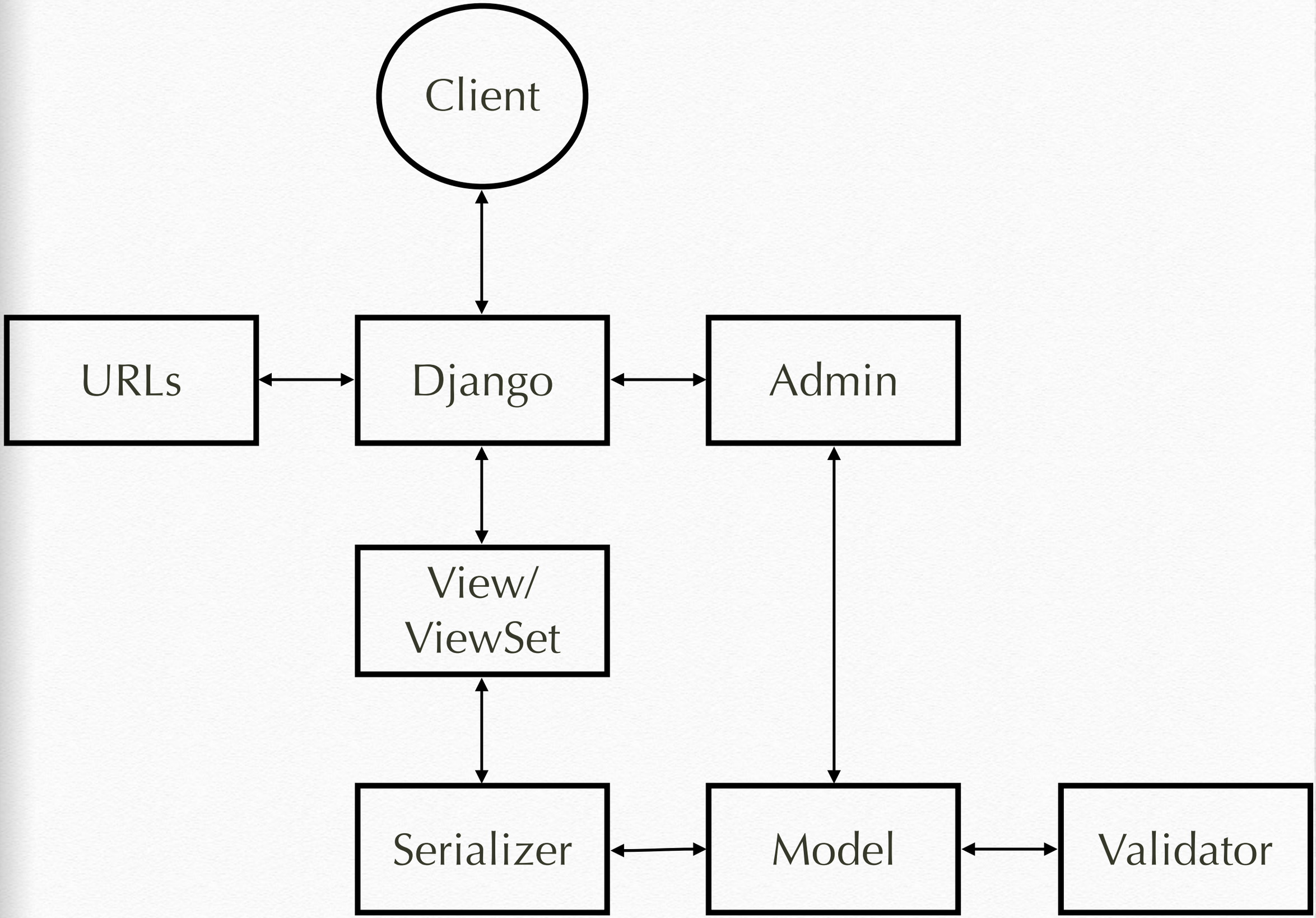
Code from Section 1

http://github.com/kennyncsu/drf-restaurant-api/blob/master/section1.zip

BREAK

Code from Section 1

http://github.com/kennyncsu/drf-restaurant-api/blob/master/section1.zip


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

Benefits of Admin Interface

- Built-In
- Immediately Available
- Customizable
- Supports Permissions
- Potential User Interface

Build Menu Admin admin.py

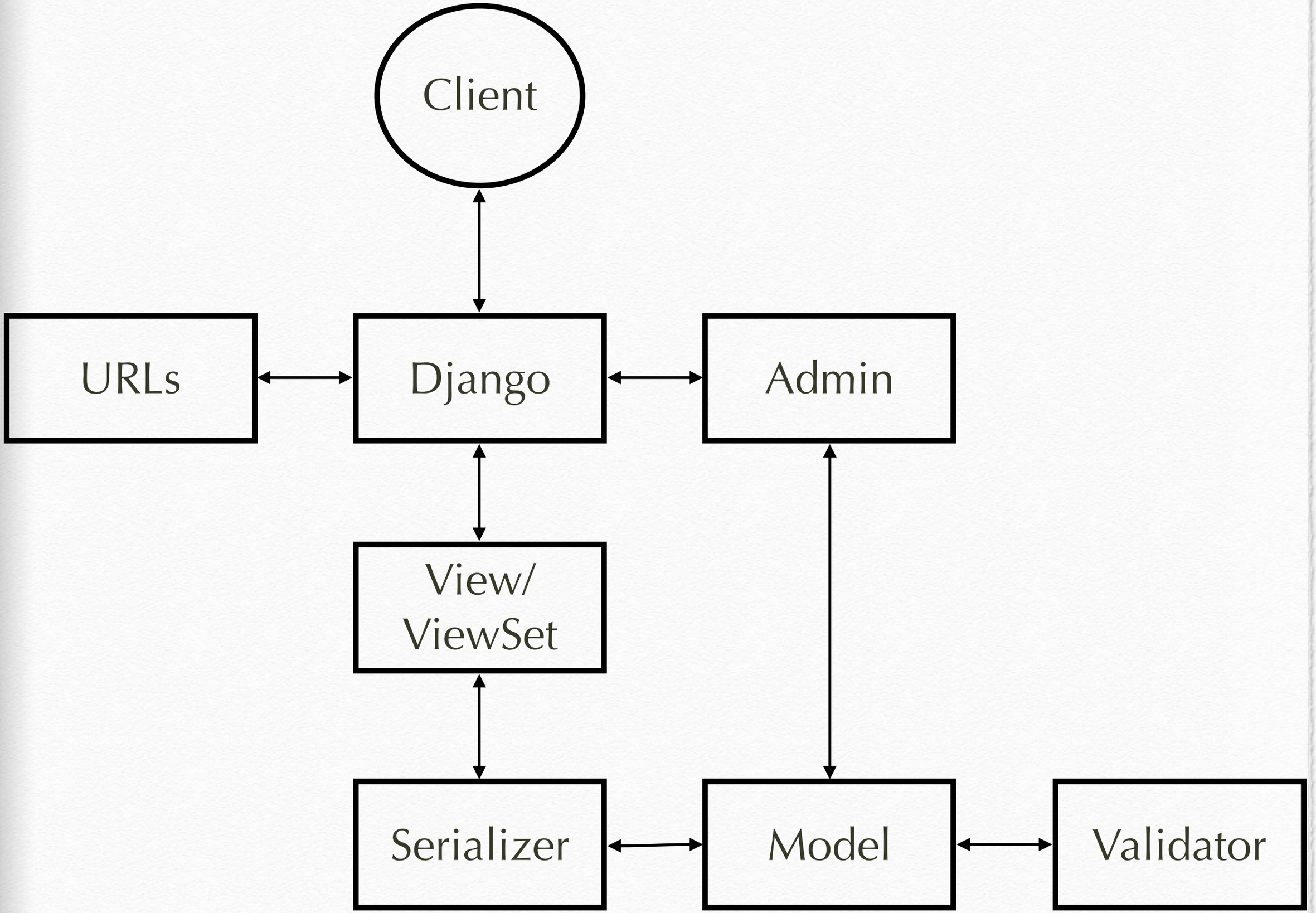
- Default Admin

```python
from django.contrib import admin
from tutorial.restaurantapi.models import Menu, MenuItem

# Register your models here.


class MenuAdmin(admin.ModelAdmin):

    pass


class MenuItemAdmin(admin.ModelAdmin):

    pass


admin.site.register(Menu, MenuAdmin)
admin.site.register(MenuItem, MenuItemAdmin)
```

Build Menu Admin admin.py

- Add Menu Fields to Admin List View
- Add MenuItem Fields to Admin List View

```
# Register your models here.

class MenuAdmin(admin.ModelAdmin):

    list_display = ['id', 'name', 'description', 'chef', 'available']


class MenuItemAdmin(admin.ModelAdmin):

    list_display = ['id', 'name', 'description', 'cost_to_make',
        'sale_price', 'available', 'menu']


admin.site.register(Menu, MenuAdmin)
admin.site.register(MenuItem, MenuItemAdmin)
```


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

TIME FOR REST (API)?

Modify Settings settings.py

- Add DRF to Installed Apps

```
32  INSTALLED_APPS = (
33      'django.contrib.admin',
34      'django.contrib.auth',
35      'django.contrib.contenttypes',
36      'django.contrib.sessions',
37      'django.contrib.messages',
38      'django.contrib.staticfiles',
39      'rest_framework',
40      'tutorial.restaurantapi',
41  )
```

Serializers[3]

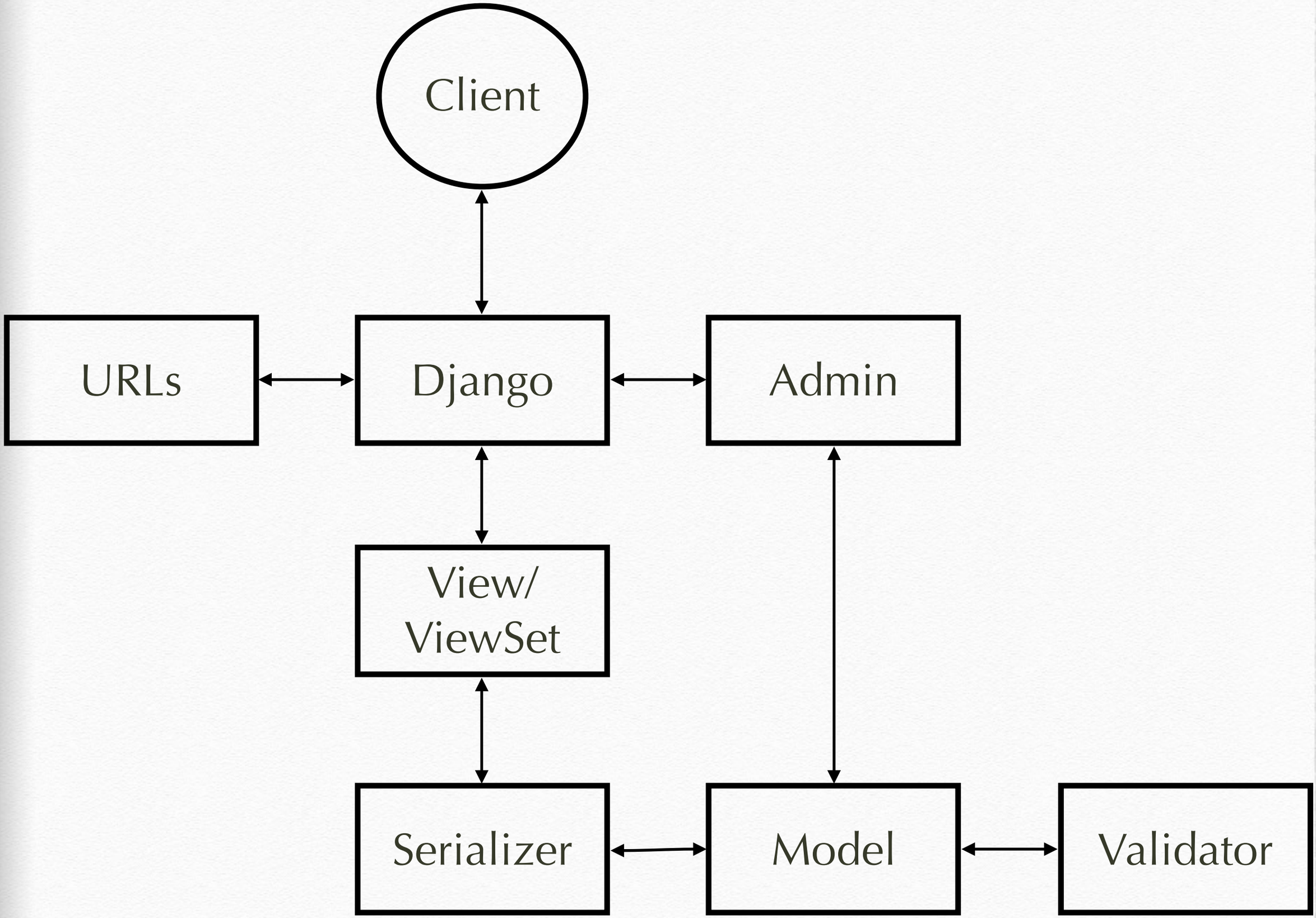
- Complex Data to/from Python Datatypes
- Generically Represent Output Data
  - JSON
  - XML
- Validate Incoming Data
- Customizable

Model Serializers[3]

- Automatically Generate Model Fields
- Automatically Generate Validators
- Default Functions:
  - Create New Object
  - Update Existing Object
- HyperLinked Model Serializer
  - HyperLink Represents Relationship

Create Serializers serializers.py

```
1  from django.contrib.auth.models import User, Group
2  from models import Menu, MenuItem
3  from rest_framework import serializers
4
5  class UserSerializer(serializers.HyperlinkedModelSerializer):
6      class Meta:
7          model = User
8          fields = ('url', 'username', 'first_name', 'last_name',
9              'email', 'is_staff', 'groups')
10
11 class GroupSerializer(serializers.HyperlinkedModelSerializer):
12     class Meta:
13         model = Group
14         fields = ('url', 'name')
15
16 class MenuSerializer(serializers.HyperlinkedModelSerializer):
17     class Meta:
18         model = Menu
19         fields = ('url', 'name', 'description', 'chef', 'available')
20
21 class MenuItemSerializer(serializers.HyperlinkedModelSerializer):
22     class Meta:
23         model = MenuItem
24         fields = ('url', 'name', 'description', 'cost_to_make',
25             'sale_price', 'available', 'menu')
```

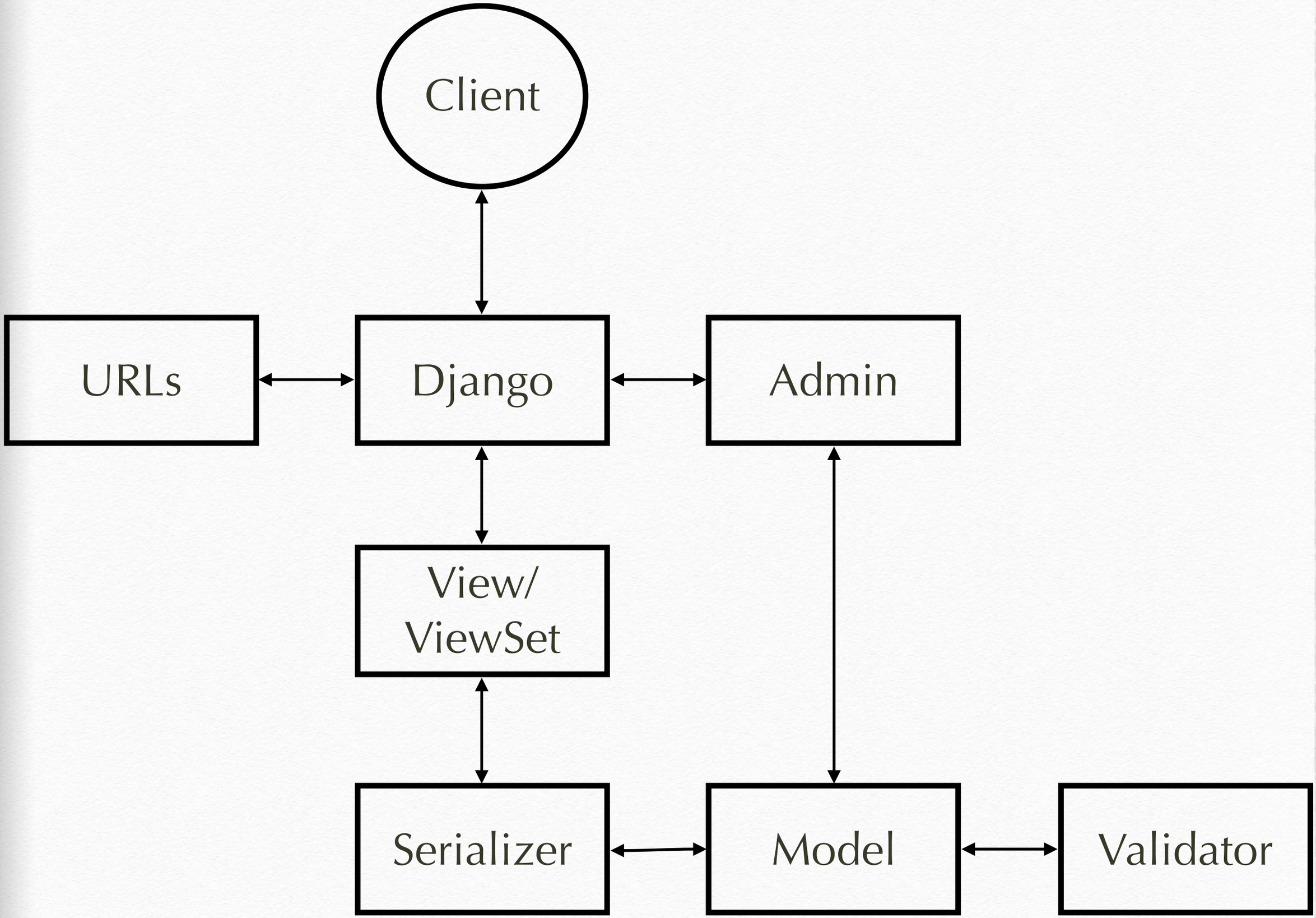

Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

ViewSets

- Single Class for Set of Related Views
- Predefined or Build Your Own
- Read vs Get
- Update vs Put
- Minimize Coding
- Less Explicit

Add Viewsets
views.py

```
1  from django.contrib.auth.models import User, Group
2  from django.shortcuts import render
3  from models import Menu, MenuItem
4  from rest_framework import viewsets
5  from serializers import *
6
7  # Create your views here.
8
9  class UserViewSet(viewsets.ModelViewSet):
10
11     queryset = User.objects.all()
12     serializer_class = UserSerializer
13
14 class GroupViewSet(viewsets.ModelViewSet):
15
16     queryset = Group.objects.all()
17     serializer_class = GroupSerializer
18
19 class MenuViewSet(viewsets.ModelViewSet):
20
21     queryset = Menu.objects.all()
22     serializer_class = MenuSerializer
23
24 class MenuItemViewSet(viewsets.ModelViewSet):
25
26     queryset = MenuItem.objects.all()
27     serializer_class = MenuItemSerializer
```

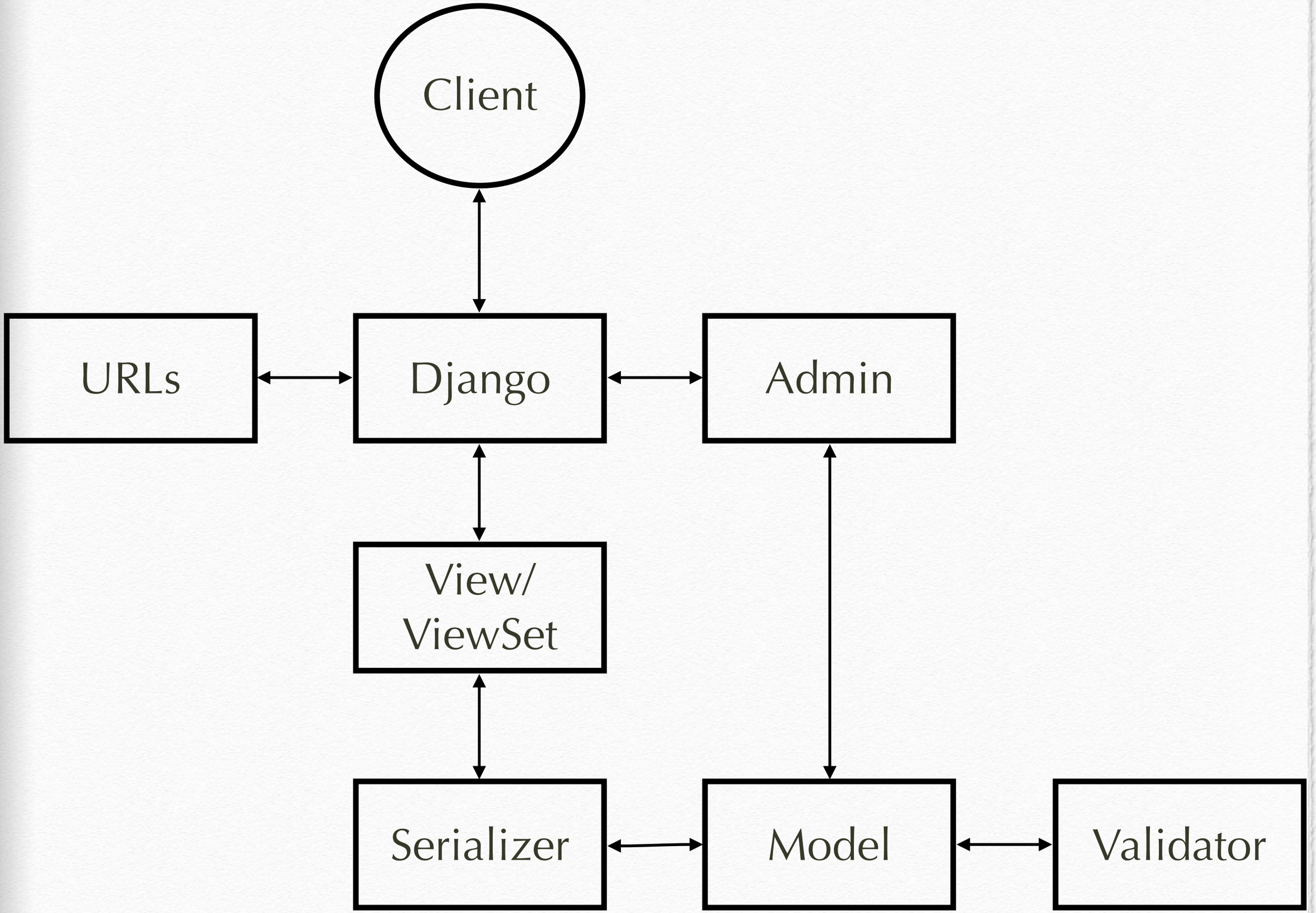

Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

Routers for URLs

- Routers
  - Register ViewSets
    - Automatically Handles URL Config.
- DefaultRouter
  - Customizable

Add Router to URLs
urls.py

```python
from django.conf.urls import patterns, include, url
from django.contrib import admin
from rest_framework import routers
from restaurantapi.views import *

router = routers.DefaultRouter()
router.register(r'user', UserViewSet)
router.register(r'group', GroupViewSet)
router.register(r'menu', MenuViewSet)
router.register(r'menuitem', MenuItemViewSet)

urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'tutorial.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    url(r'^', include(router.urls)),
    url(r'^admin/', include(admin.site.urls)),
    url(r'api-auth/', include('rest_framework.urls', namespace='rest_framework')),
)
```


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

TRY IT OUT

Usages

- Browsable API
  - Customizable via Serializers
- CURL
- External Services

Using CURL

- Get List of Services
  - curl http://localhost:8000/
- Get List of Menus
  - curl http://localhost:8000/menu/

What's Next?

- No Permissions
- Unable to Set User Password
- No Validation
- Enable XML
- Public API

Adding API Permissions

- Built-In Permission
  - Entire API
  - Specific API Endpoints
- Custom Permission Check

Permission for Entire API settings.py

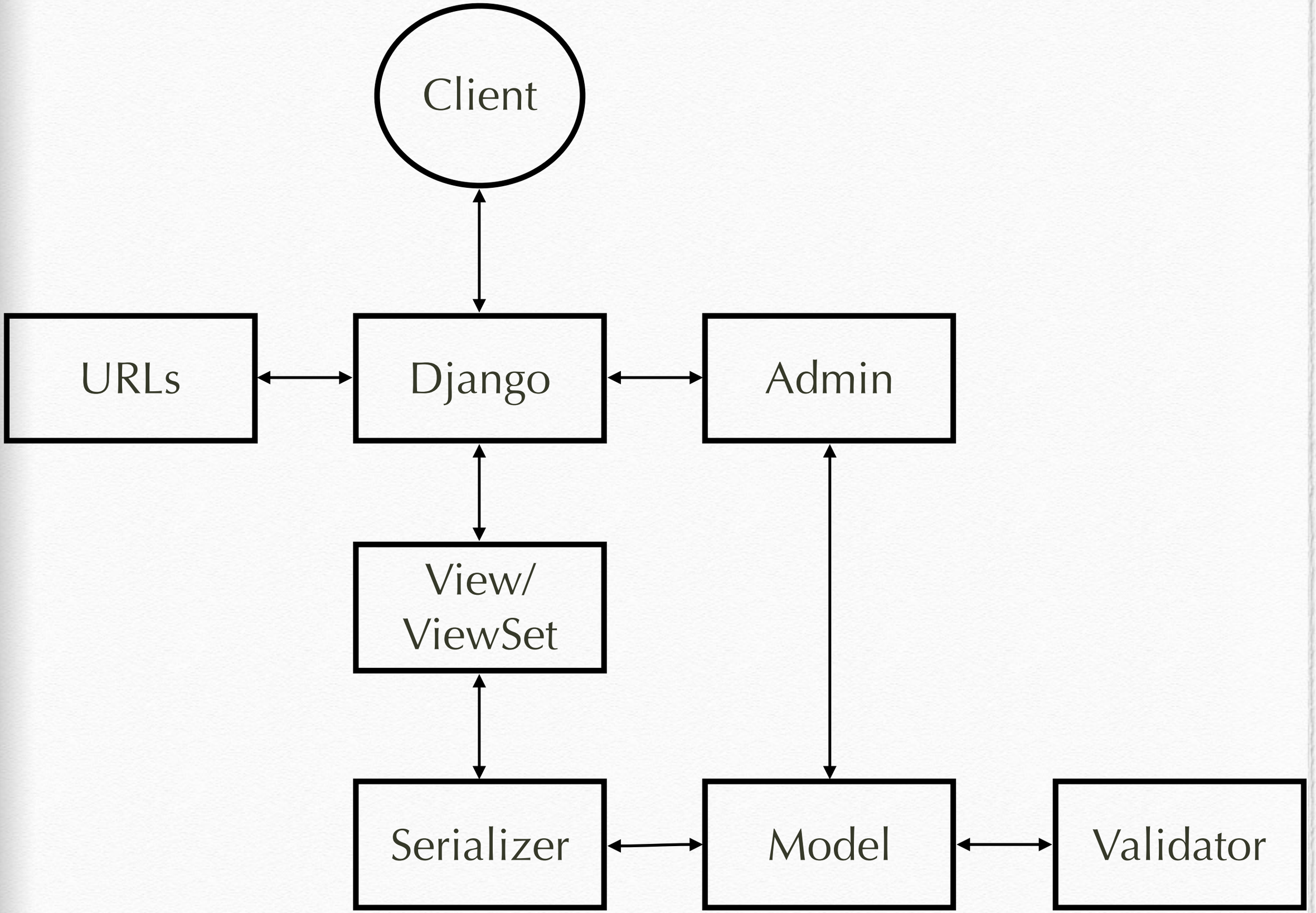
- Require Authenticated User

```
43  REST_FRAMEWORK = {
44      'DEFAULT_PERMISSION_CLASSES': ('rest_framework.permissions.IsAuthenticated',),
45  }
```

Permission for Specific API views.py

- Restrict Access to Users, Groups

```
1  from django.contrib.auth.models import User, Group
2  from django.shortcuts import render
3  from models import Menu, MenuItem
4  from rest_framework import viewsets
5  from rest_framework.permissions import IsAdminUser
6  from serializers import *
7
8  # Create your views here.
9
10 class UserViewSet(viewsets.ModelViewSet):
11
12     permission_classes = (IsAdminUser,)
13
14     queryset = User.objects.all()
15     serializer_class = UserSerializer
16
17 class GroupViewSet(viewsets.ModelViewSet):
18
19     permission_classes = (IsAdminUser,)
20
21     queryset = Group.objects.all()
22     serializer_class = GroupSerializer
```


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

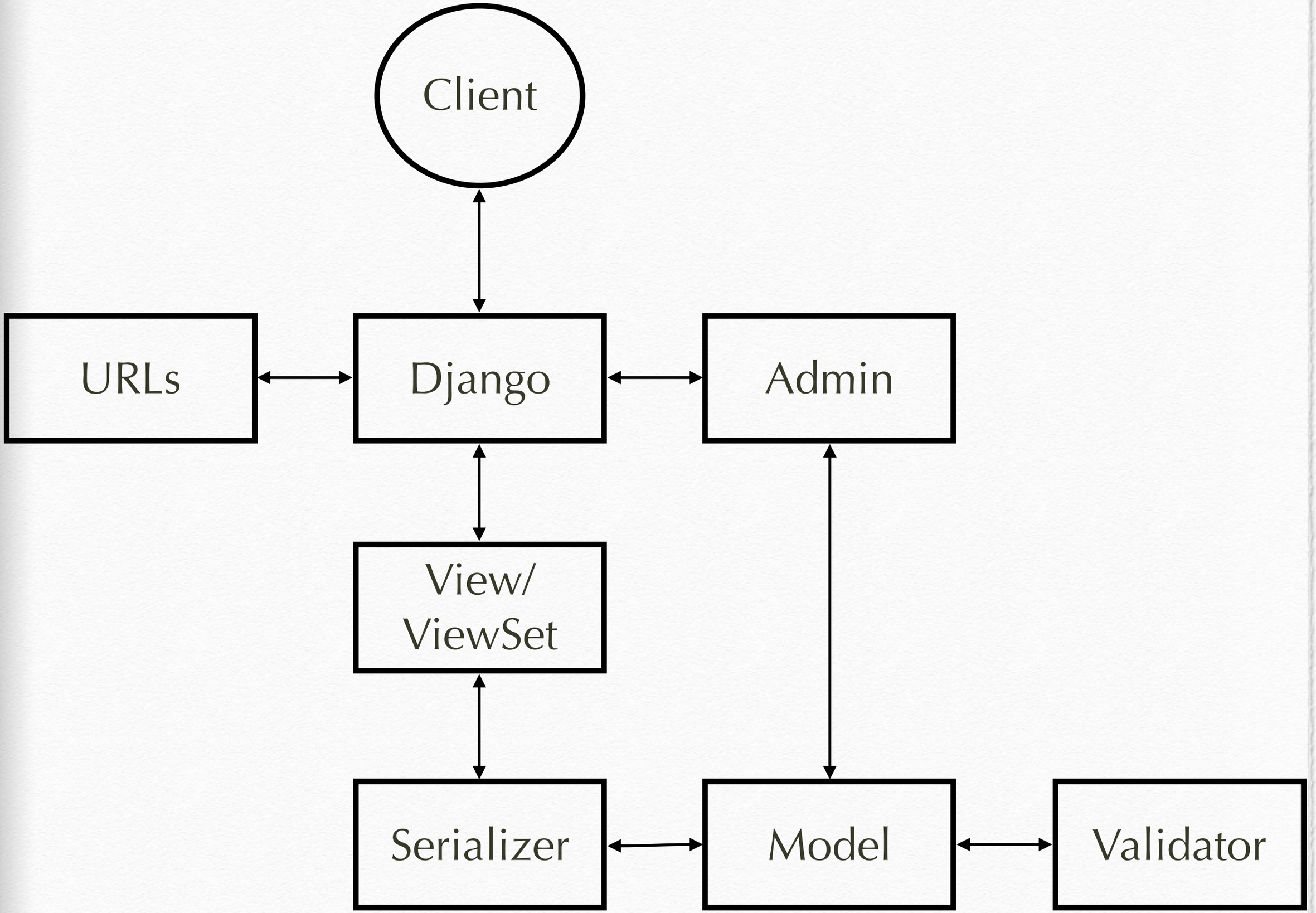
Setting Password

- Default Encryption
- Enable Default Password

Set Default Password serializers.py

- Update UserSerializer

```
    def create(self, validated_data):
        # override create to set a default password

        if not 'is_staff' in validated_data:
            validated_data['is_staff'] = False

        user = User(
            email=validated_data['email'],
            username=validated_data['username'],
            is_staff=validated_data['is_staff'],
            first_name=validated_data['first_name'],
            last_name=validated_data['last_name'],
        )

        user.set_password(validated_data['username'])
        user.save()

        for group in validated_data['groups']:
            user.groups.add(group)

        return user
```


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

Validation

- Validation
  - Forms
  - Models
  - Serializers
  - Views/ViewSets

Menu Item Validation

- Validate Slug Field for Name (Built-In)
  - Letters
  - Numbers
  - Underscores
  - Hyphens
- Validate Limit on Cost to Make (Custom)

Custom Validator validators.py

- Validate Cost to Make Limit

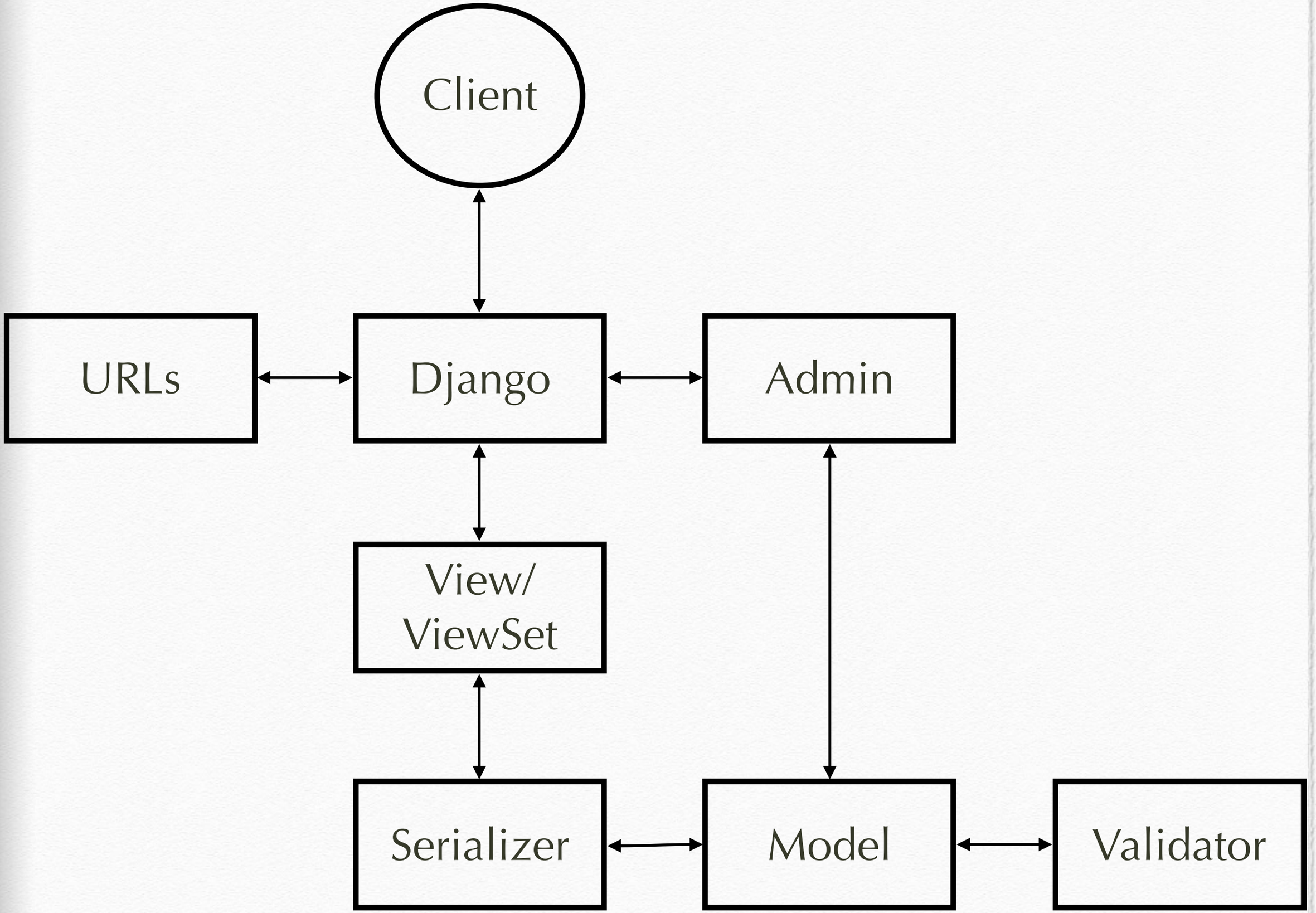
```
1
2  from django.core.exceptions import ValidationError
3
4  def validate_acceptable_cost_to_make(value):
5
6      if value > 5.00:
7
8          raise ValidationError('Unable to accept items that cost ' + \
9              'more than $5.00 to make.')
```

Menu Item Validation models.py

- Validate Slug Field for Name
- Validate Limit on Cost to Make

```
4  from django.core.validators import validate_slug
5  from validators import validate_acceptable_cost_to_make
```

```
20 class MenuItem(models.Model):
21
22     name = models.CharField(max_length=32, unique=True,
23         validators=[validate_slug])
24     description = models.CharField(max_length=200)
25     cost_to_make = models.DecimalField(decimal_places=2, max_digits=5,
26         validators=[validate_acceptable_cost_to_make])
27     sale_price = models.DecimalField(decimal_places=2, max_digits=5)
28     available = models.BooleanField(default=False)
29     menu = models.ForeignKey(Menu)
30
31     def __unicode__(self):
32         return u'%s' % (self.name)
```


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

Renderers[3]

- Return Response of Specific Media Type
- Content Negotiation:
  - Accept Header
  - URL Suffix such as .json
- Built-In or Build Your Own
- Enable Globally or Per View

Enable XML Renderer settings.py

```
43  REST_FRAMEWORK = {
44      'DEFAULT_PERMISSION_CLASSES': ('rest_framework.permissions.IsAuthenticated',),
45      'DEFAULT_RENDERER_CLASSES': (
46          'rest_framework.renderers.JSONRenderer', # default renderer
47          'rest_framework.renderers.BrowsableAPIRenderer', # default renderer
48          'rest_framework.renderers.XMLRenderer',
49      ),
50  }
```

Public API

- Support GET Action
- Customization Options
  - ViewSet
    - API Views
    - Mixins for List, Create, Etc.
- Set No Permissions
- Show All Available Menus

Using Mixins
views.py, urls.py

- Return List of Available Menus

```
 8  # for custom GET of available menu list
 9  from rest_framework import mixins
10  from rest_framework import generics
11
12  # Create your views here.
13
14  class AvailableMenuList(mixins.ListModelMixin, generics.GenericAPIView):
15
16      permission_classes = ()
17      queryset = Menu.objects.filter(available=True)
18      serializer_class = MenuSerializer
19
20      def get(self, request, *args, **kwargs):
21
22          return self.list(request, *args, **kwargs)
```

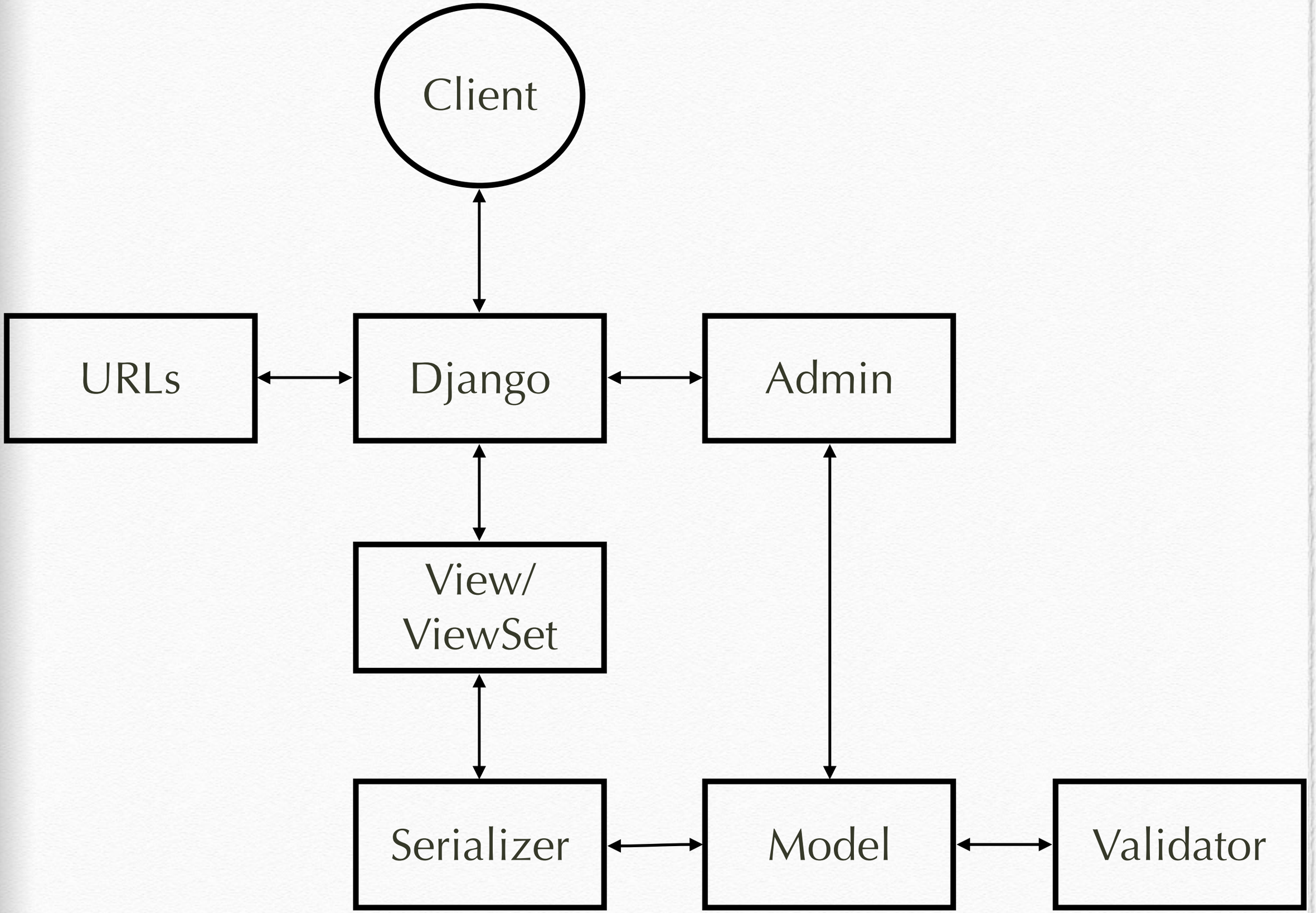
```
17      url(r'^', include(router.urls)),
18      url(r'^admin/', include(admin.site.urls)),
19      url(r'api-auth/', include('rest_framework.urls', namespace='rest_framework')),
20      url(r'^available_menus/$', AvailableMenuList.as_view()),
21  )
```

Using API View: views.py

```
11  from django.http import Http404
12  from rest_framework.exceptions import APIException
13  from rest_framework.response import Response
14  from rest_framework.views import APIView
15
16  class ForbiddenAccess(APIException):
17      status_code = 403
18      default_detail = 'Action Forbidden'
19
20  class AvailableMenuDetail(APIView):
21
22      permission_classes = ()
23
24      def get_object(self, pk):
25          try:
26              return Menu.objects.get(pk=pk, available=True)
27
28          except Menu.DoesNotExist:
29              raise Http404
30
31      def get(self, request, pk, format=None):
32          menu = self.get_object(pk)
33
34          serializer = MenuSerializer(menu, context={'request':request})
35
36          return Response(serializer.data)
37
38      def put(self, request, pk, format=None):
39
40          raise ForbiddenAccess
41
42      def delete(self, request, pk, format=None):
43
44          raise ForbiddenAccess
```

Using API View
urls.py

```
17      url(r'^', include(router.urls)),
18      url(r'^admin/', include(admin.site.urls)),
19      url(r'api-auth/', include('rest_framework.urls', namespace='rest_framework')),
20      url(r'^available_menus/$', AvailableMenuList.as_view()),
21      url(r'^available_menus/(?P<pk>[0-9]+)/$', AvailableMenuDetail.as_view()),
22  )
```


Client
URLs
Django
Admin
View/
ViewSet
Serializer
Model
Validator

Code from Section 2

http://github.com/kennyncsu/drf-restaurant-api/blob/master/section2.zip

Exploration

- Enhance GET of Available Menus
  - Include Check on Available Menu Items
- Add Ingredients for Menu Items
  - Add Availability
- Add Additional Groups
- Use More Validators
- Build More Views

References

1. Representational State Transfer
2. Django
3. Django REST Framework

Thank you!

Email: kenny.yarboro@gmail.com

http://github.com/kennyncsu/drf-restaurant-api