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

![django]

- High-level Python Web Framework

- Quickly Develop Applications

- Built-In Security

- Scalable

- Built-In Administrative Interface

django

- ❖ MVC vs MTV

- ❖ Model = Model

- ❖ View = Template

- ❖ Controller = View

django REST framework

- ❖ 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
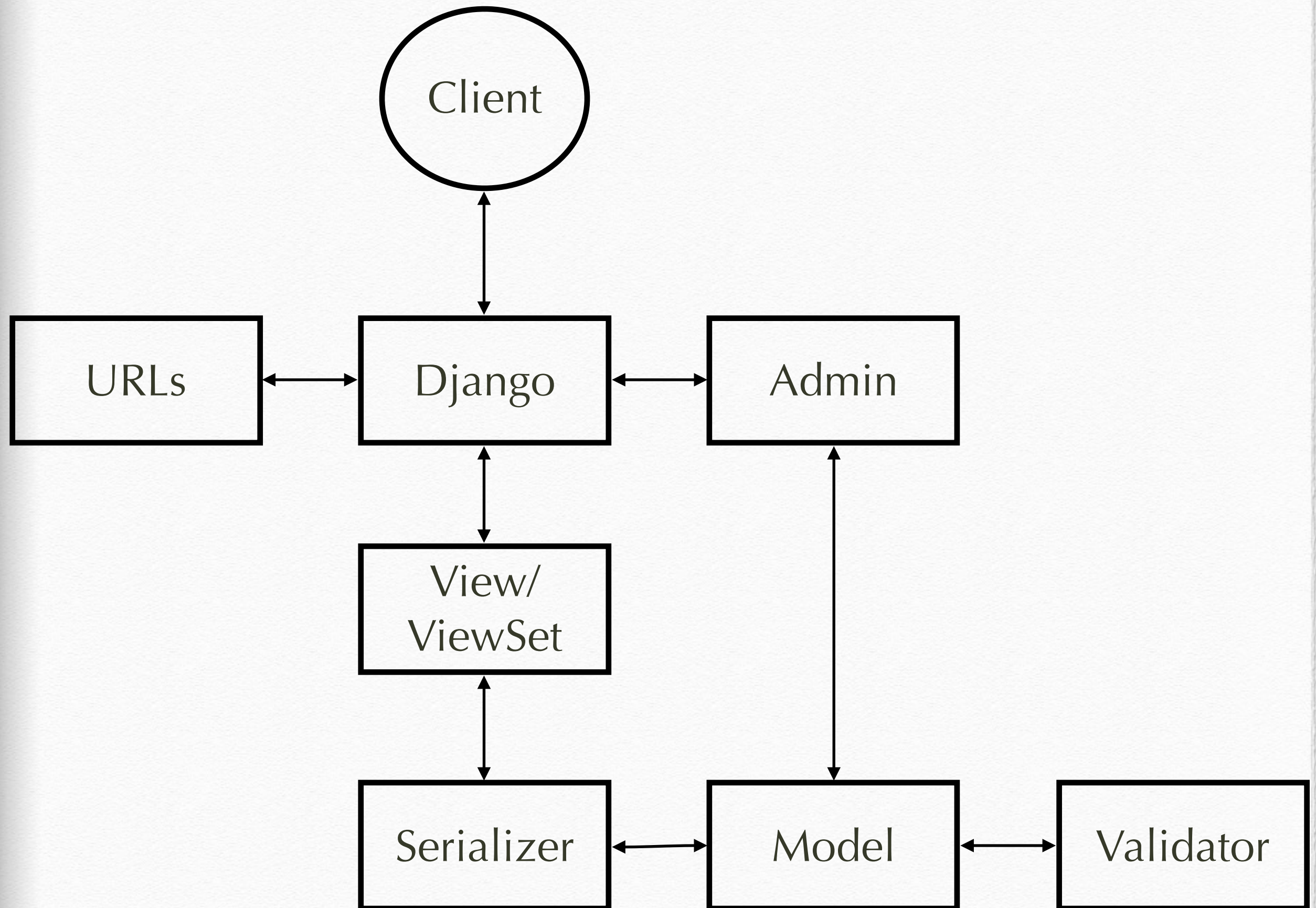
    - Windows: venv\Scripts\activate

  - deactivate

Necessary Setup

- Django

    - pip install django==1.8.4

- Django REST Framework (DRF)

    - pip install djangorestframework==3.2.2

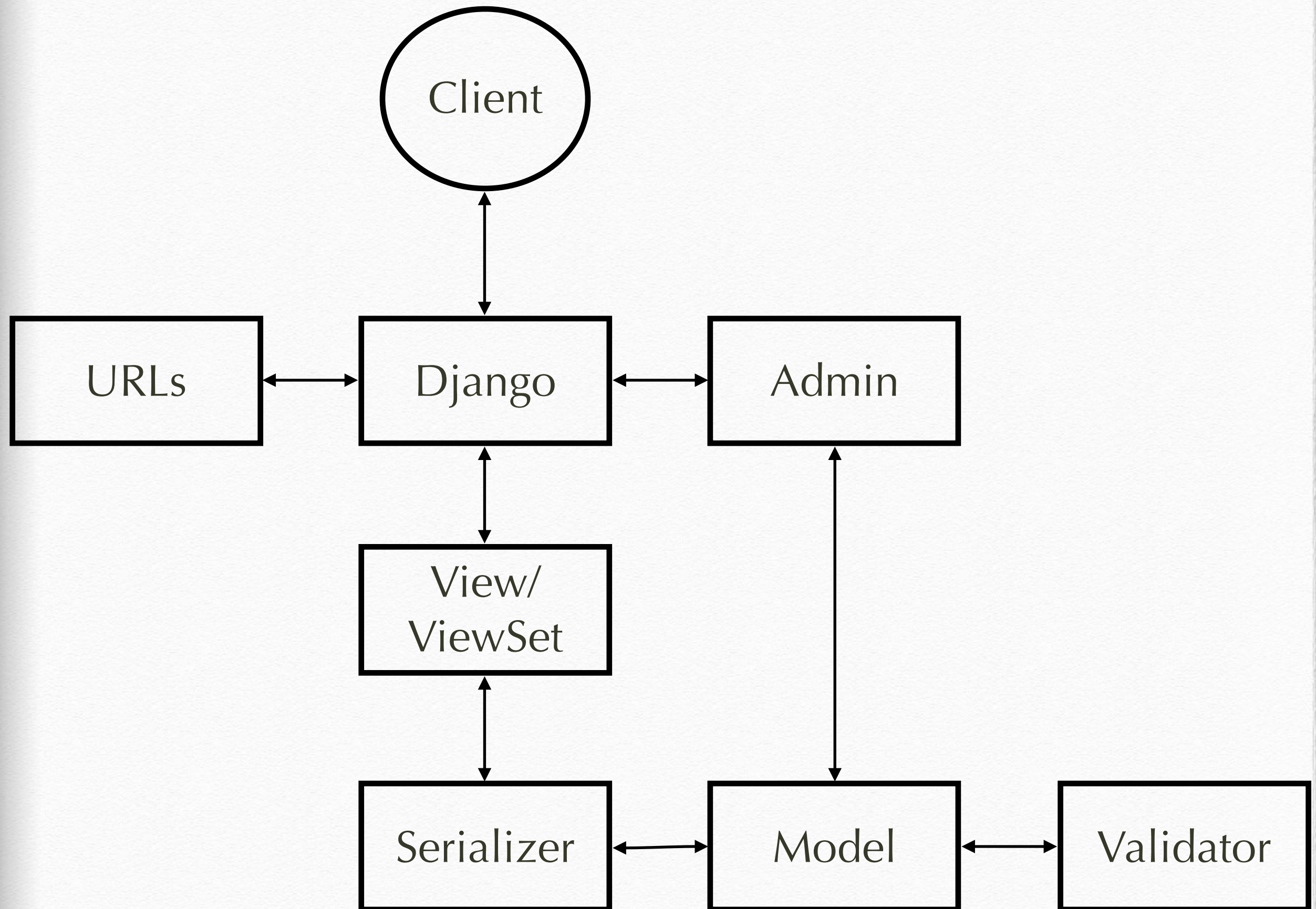# Start Project

- 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

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

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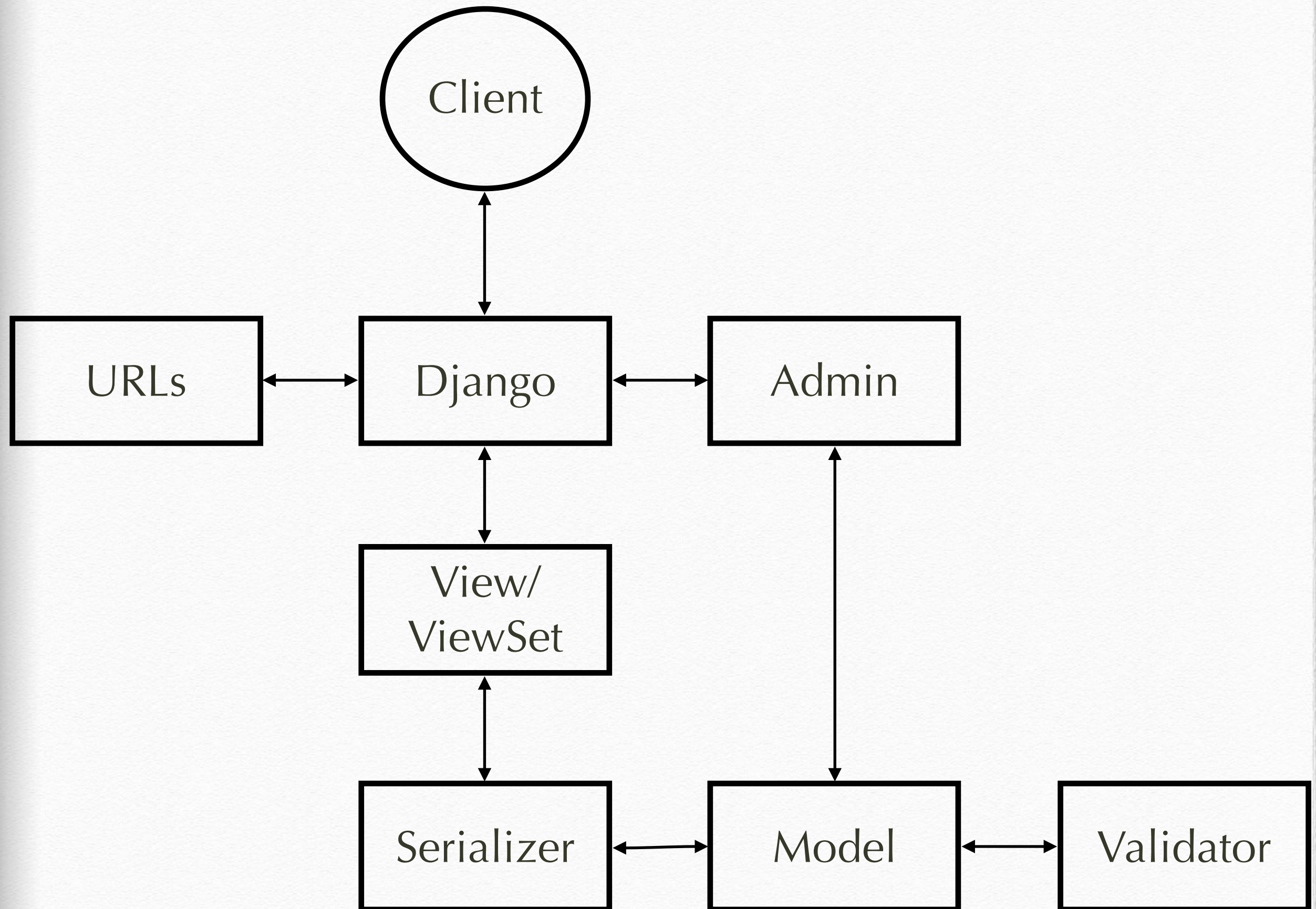
Create the Database

❖ 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

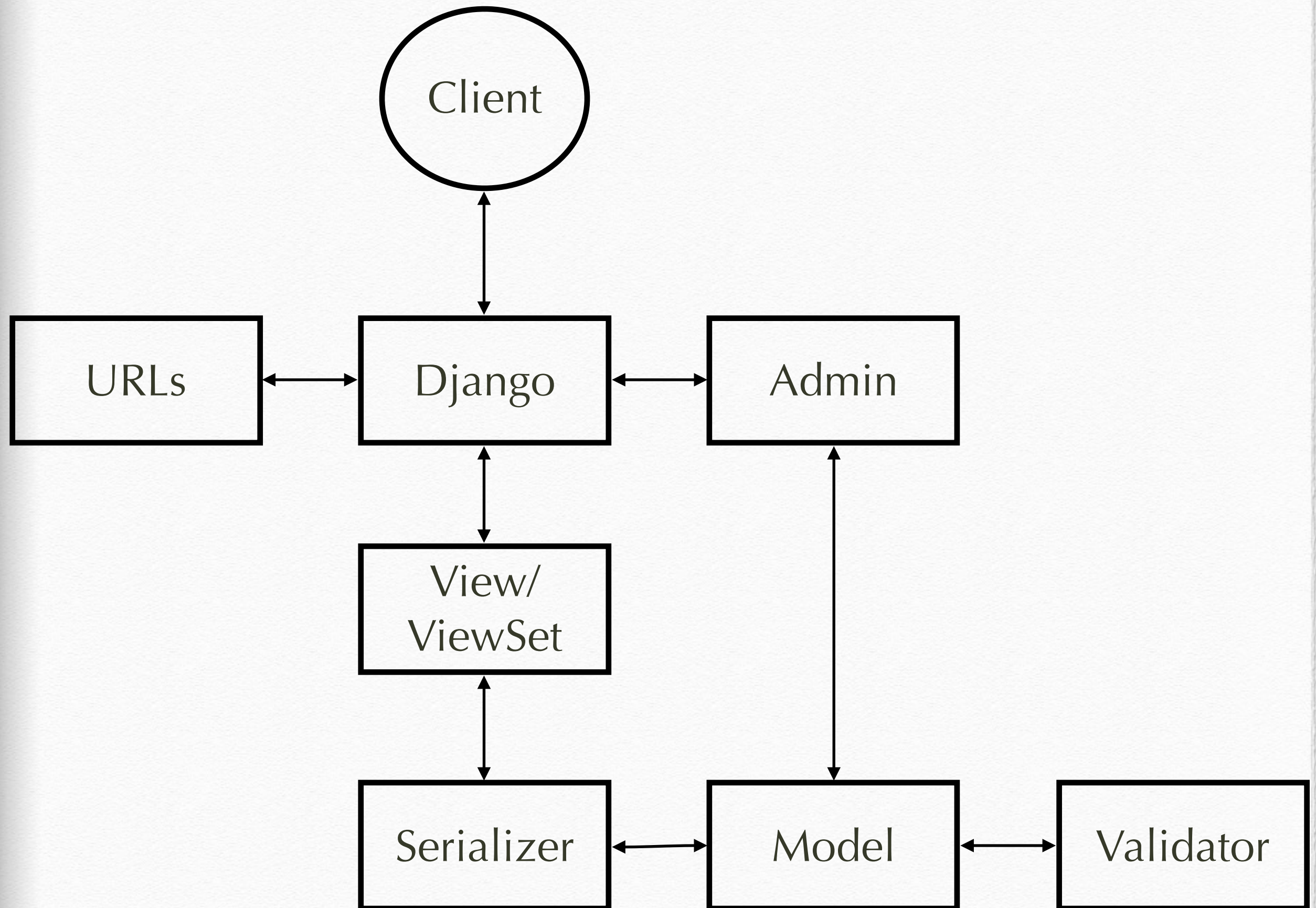❖ Create More Menu Items

❖ Create Menus

❖ Query Database for Data

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

Django ⟷ View/ViewSet

View/ViewSet ⟷ Serializer

Serializer ⟷ Model ⟷ Validator

Admin ⟷ Model

Benefits of Admin Interface

❖ Built-In

❖ Immediately Available

❖ Customizable

❖ Supports Permissions

❖ Potential User Interface
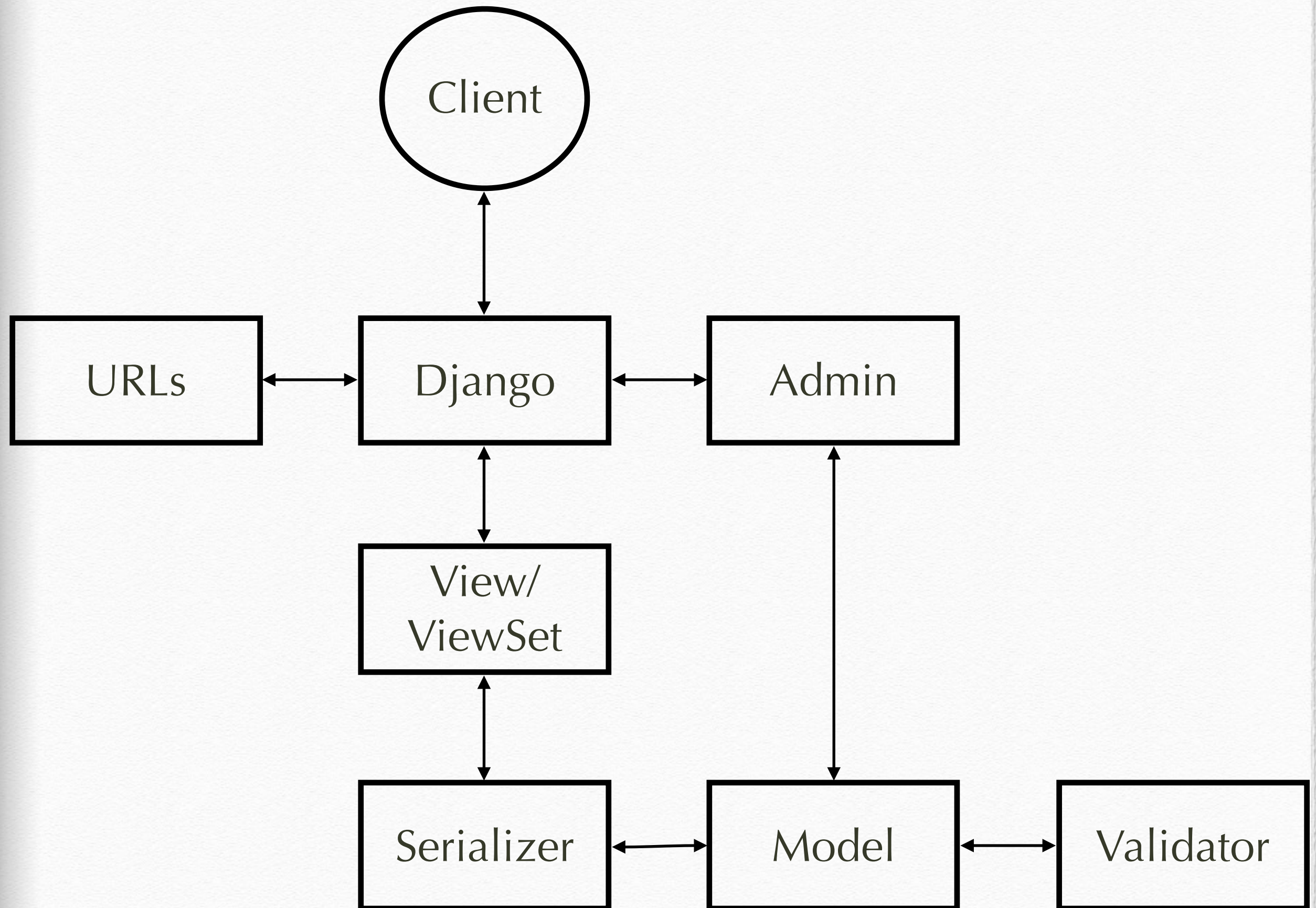
34

Build Menu Admin admin.py

❖ Default Admin

```python
1   from django.contrib import admin
2   from tutorial.restaurantapi.models import Menu, MenuItem
3
4   # Register your models here.
5
6   class MenuAdmin(admin.ModelAdmin):
7
8       pass
9
10
11  class MenuItemAdmin(admin.ModelAdmin):
12
13      pass
14
15
16  admin.site.register(Menu, MenuAdmin)
17  admin.site.register(MenuItem, MenuItemAdmin)
```

Build Menu Admin admin.py

- Add Menu Fields to Admin List View

- Add MenuItem Fields to Admin List View

```python
4   # Register your models here.
5
6   class MenuAdmin(admin.ModelAdmin):
7
8       list_display = ['id', 'name', 'description', 'chef', 'available']
9
10
11  class MenuItemAdmin(admin.ModelAdmin):
12
13      list_display = ['id', 'name', 'description', 'cost_to_make',
14              'sale_price', 'available', 'menu']
15
16
17  admin.site.register(Menu, MenuAdmin)
18  admin.site.register(MenuItem, MenuItemAdmin)
```

TIME FOR REST (API)?

Modify Settings settings.py

❖ Add DRF to Installed Apps

```
33   INSTALLED_APPS = (
34       'django.contrib.admin',
35       'django.contrib.auth',
36       'django.contrib.contenttypes',
37       'django.contrib.sessions',
38       'django.contrib.messages',
39       'django.contrib.staticfiles',
40       'rest_framework',
41       'tutorial.restaurantapi',
42   )
```

Serializers[3]

- Complex Data to/from Python Datatypes

- Generically Represent Output Data

  - JSON
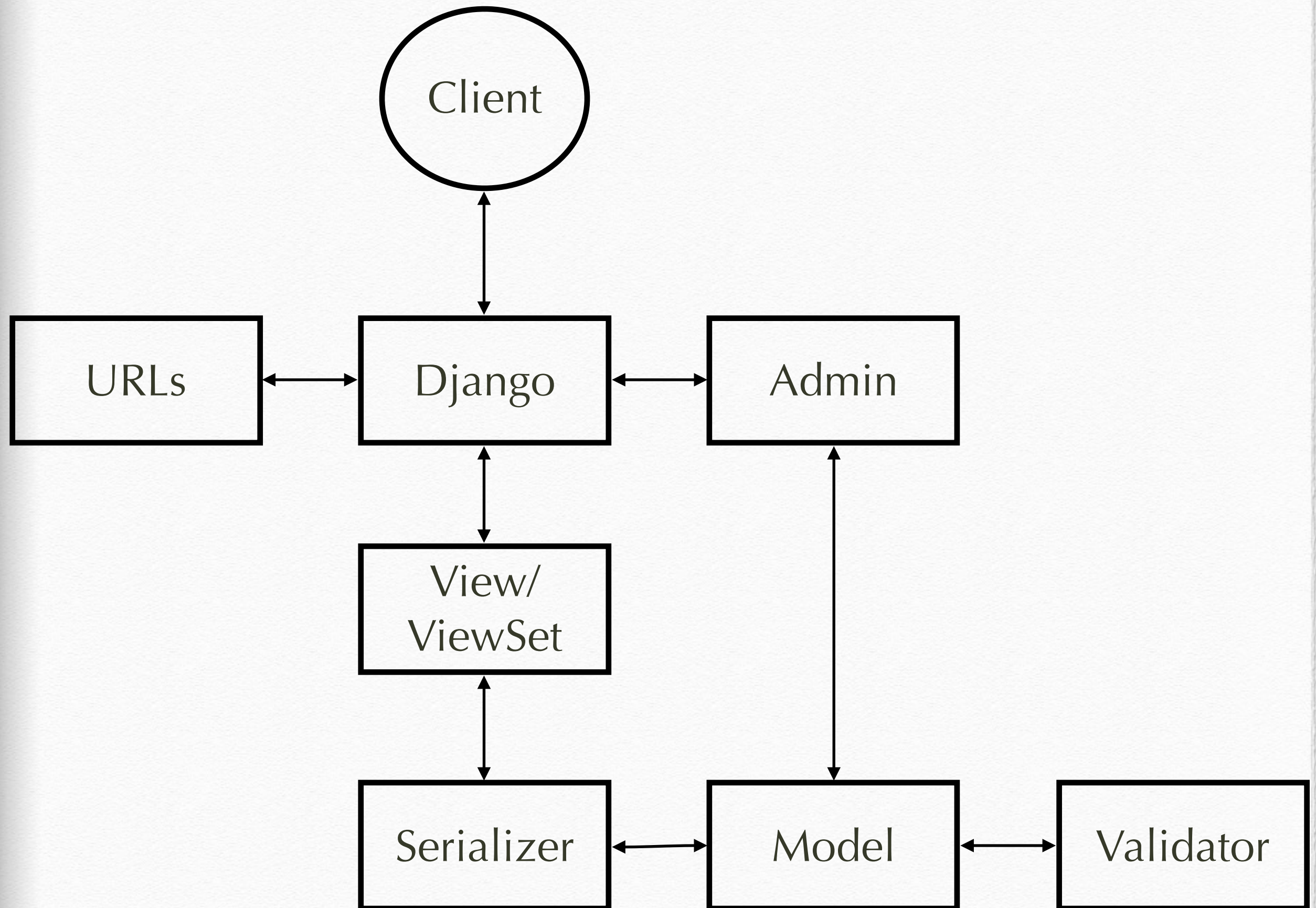
  - XML

- 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

```python
from django.contrib.auth.models import User, Group
from models import Menu, MenuItem
from rest_framework import serializers

class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ('url', 'username', 'first_name', 'last_name',
            'email', 'is_staff', 'groups')

class GroupSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Group
        fields = ('url', 'name')

class MenuSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Menu
        fields = ('url', 'name', 'description', 'chef', 'available')

class MenuItemSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = MenuItem
        fields = ('url', 'name', 'description', 'cost_to_make',
            'sale_price', 'available', 'menu')
```
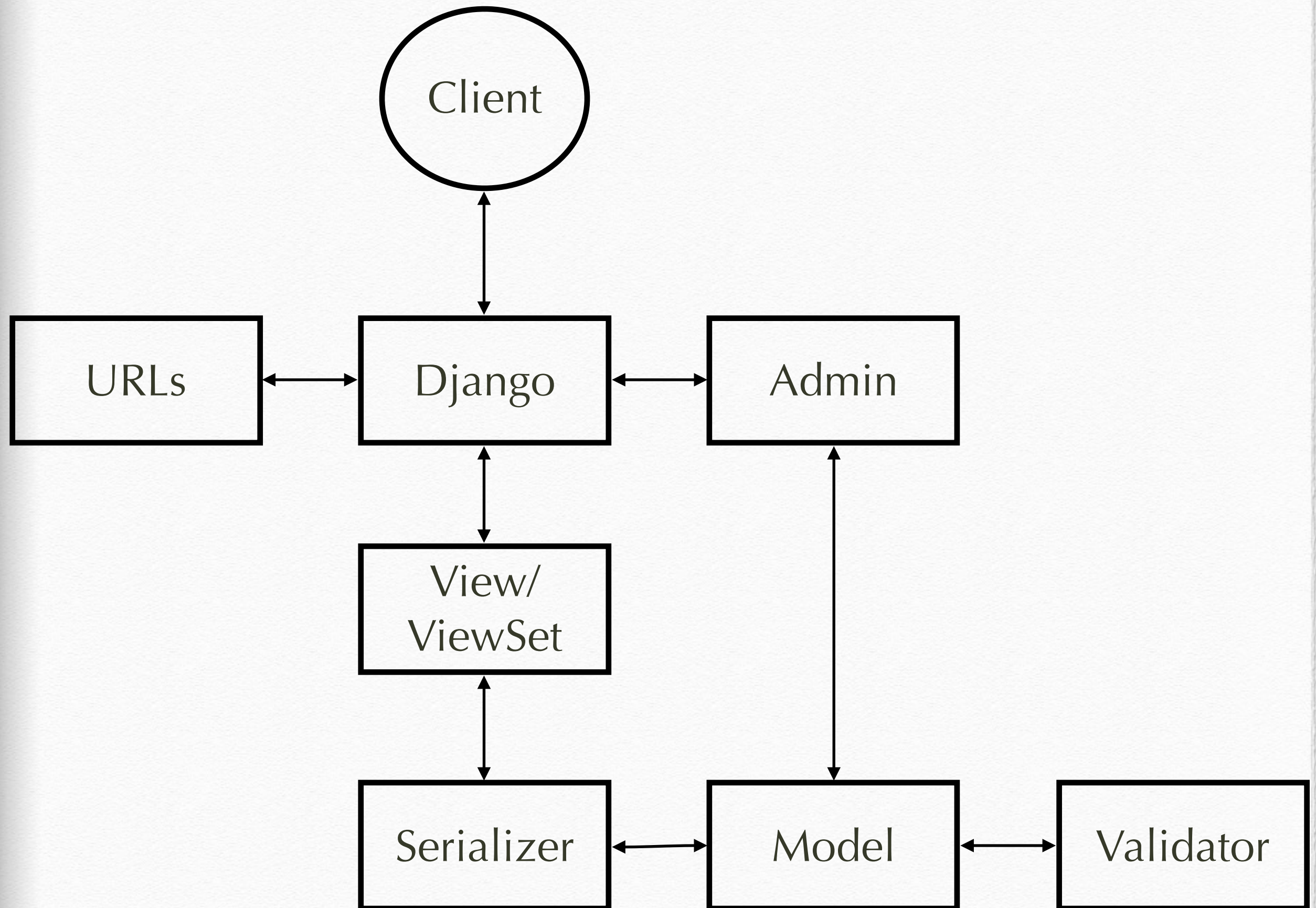
Client

URLs ↔ Django ↔ Admin

Django ↕ View/ViewSet

Admin ↕ Model

View/ViewSet ↕ Serializer

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

```python
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 UserSerializer, GroupSerializer
6  from serializers import MenuSerializer, MenuItemSerializer
7
8  # Create your views here.
9
10 class UserViewSet(viewsets.ModelViewSet):
11
12     queryset = User.objects.all()
13     serializer_class = UserSerializer
14
15 class GroupViewSet(viewsets.ModelViewSet):
16
17     queryset = Group.objects.all()
18     serializer_class = GroupSerializer
19
20 class MenuViewSet(viewsets.ModelViewSet):
21
22     queryset = Menu.objects.all()
23     serializer_class = MenuSerializer
24
25 class MenuItemViewSet(viewsets.ModelViewSet):
26
27     queryset = MenuItem.objects.all()
28     serializer_class = MenuItemSerializer
```
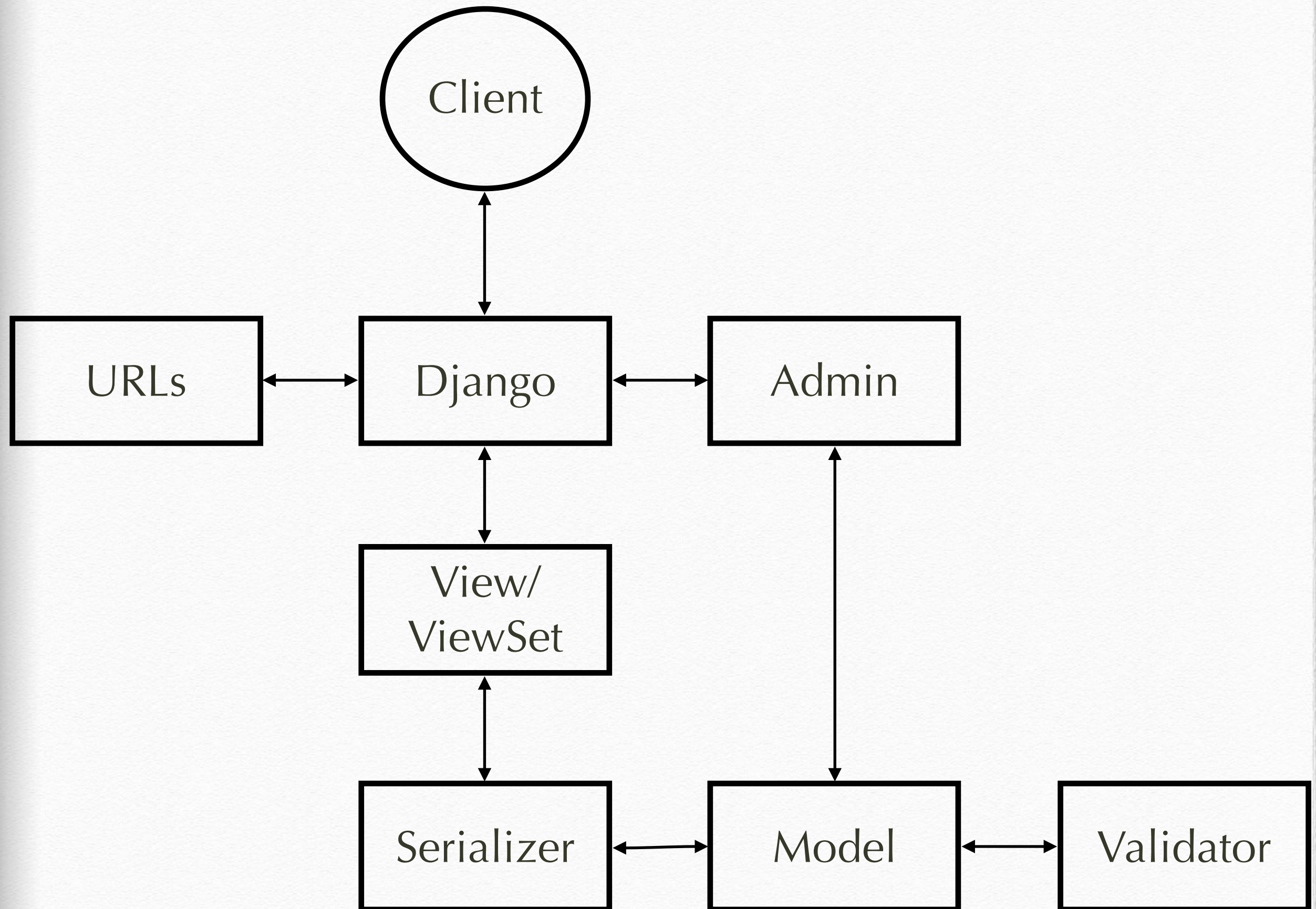
Routers for URLs

- Routers

    - Register ViewSets

    - Automatically Handles URL Config.

- DefaultRouter

    - Customizable

Add Router to URLs
urls.py

```
16  from django.conf.urls import include, url
17  from django.contrib import admin
18  from rest_framework import routers
19  from restaurantapi.views import UserViewSet, GroupViewSet
20  from restaurantapi.views import MenuViewSet, MenuItemViewSet
21
22  router = routers.DefaultRouter()
23  router.register(r'user', UserViewSet)
24  router.register(r'group', GroupViewSet)
25  router.register(r'menu', MenuViewSet)
26  router.register(r'menuitem', MenuItemViewSet)
27
28  urlpatterns = [
29      url(r'^', include(router.urls)),
30      url(r'^admin/', include(admin.site.urls)),
31      url(r'api-auth/', include('rest_framework.urls', namespace='rest_framework')),
32  ]
```

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
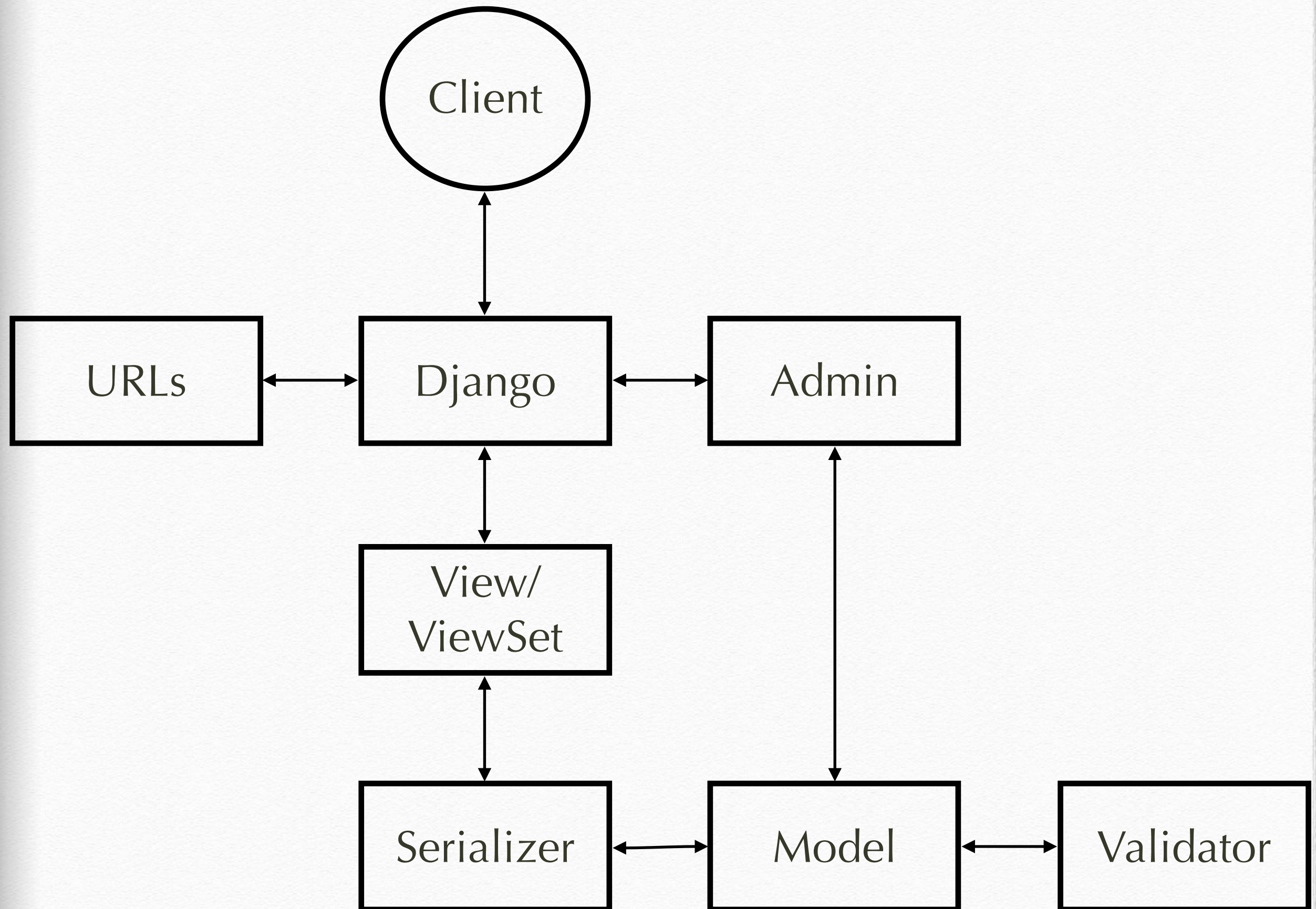
- ❖ Require Authenticated User

```
44  REST_FRAMEWORK = {
45      'DEFAULT_PERMISSION_CLASSES': ('rest_framework.permissions.IsAuthenticated',),
46  }
```

Permission for Specific API views.py

- Restrict Access to Users, Groups

```python
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 UserSerializer, GroupSerializer
7   from serializers import MenuSerializer, MenuItemSerializer
8
9   # Create your views here.
10
11  class UserViewSet(viewsets.ModelViewSet):
12
13      permission_classes = (IsAdminUser,)
14
15      queryset = User.objects.all()
16      serializer_class = UserSerializer
17
18  class GroupViewSet(viewsets.ModelViewSet):
19
20      permission_classes = (IsAdminUser,)
21
22      queryset = Group.objects.all()
23      serializer_class = GroupSerializer
```
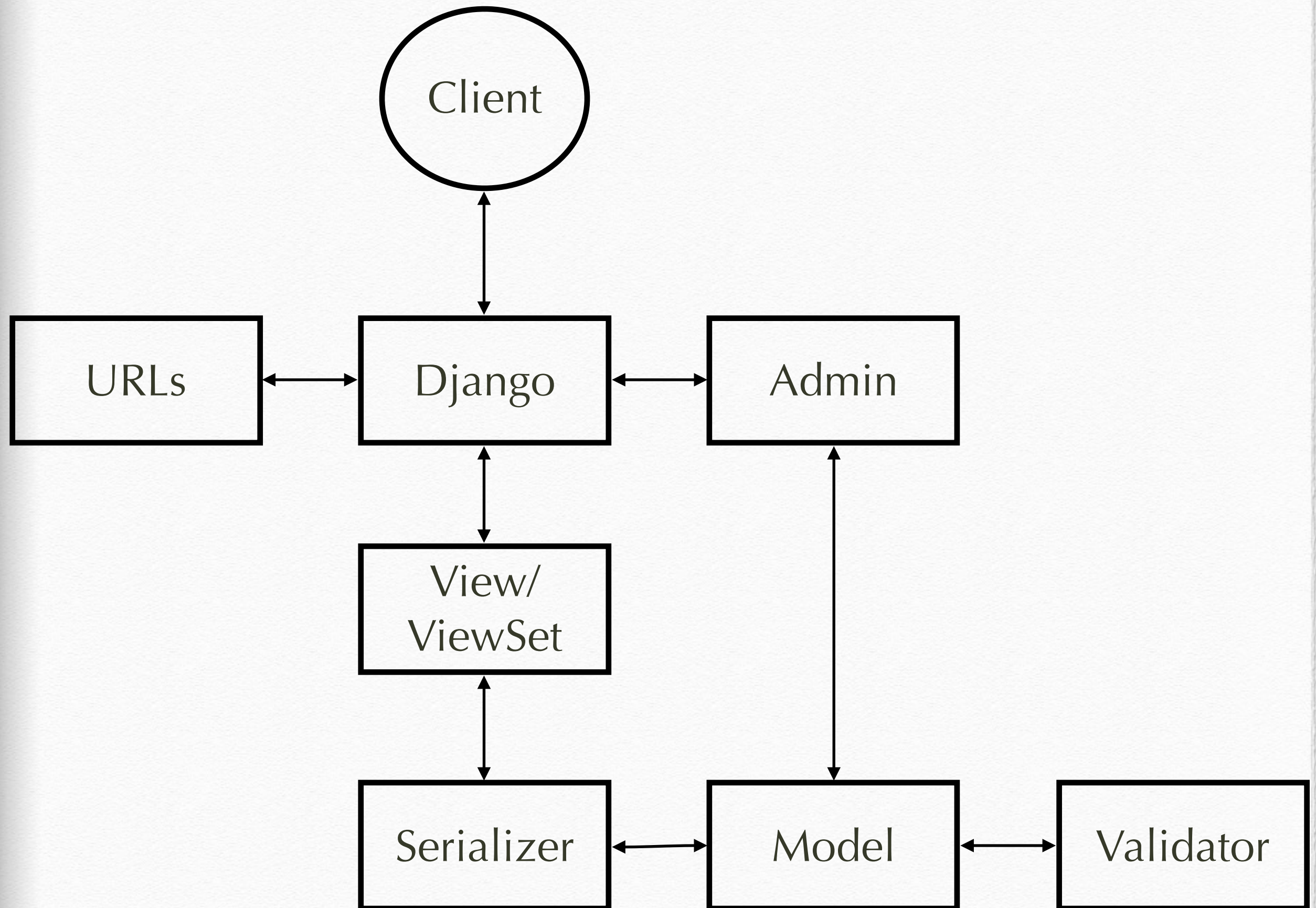
Setting Password

❖ Default Encryption

❖ Enable Default Password

Set Default Password
serializers.py

❖ Update UserSerializer

```python
 7    def create(self, validated_data):
 8        # override create to set a default password
 9
10        if not 'is_staff' in validated_data:
11            validated_data['is_staff'] = False
12
13        user = User(
14            email=validated_data['email'],
15            username=validated_data['username'],
16            is_staff=validated_data['is_staff'],
17            first_name=validated_data['first_name'],
18            last_name=validated_data['last_name'],
19        )
20
21        user.set_password(validated_data['username'])
22        user.save()
23
24        for group in validated_data['groups']:
25            user.groups.add(group)
26
27        return user
```

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

```python
from django.core.exceptions import ValidationError

def validate_acceptable_cost_to_make(value):

    if value > 5.00:

        raise ValidationError('Unable to accept items that cost ' + \
            'more than $5.00 to make.')
```
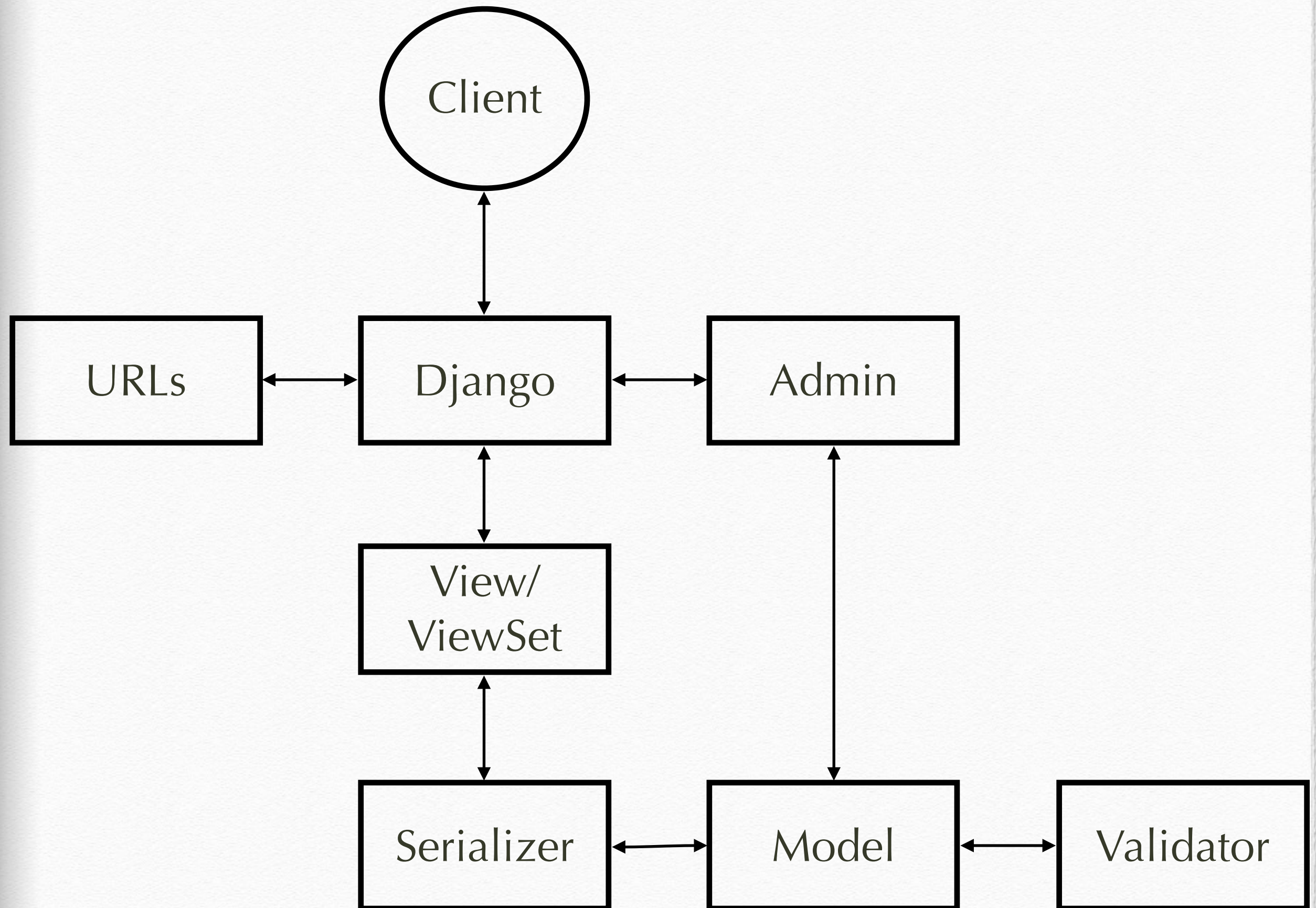
Menu Item Validation models.py

- Validate Slug Field for Name

- Validate Limit on Cost to Make

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

```python
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)
```

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

- ❖ REST Framework XML

  - ❖ pip install djangorestframework-xml==1.2.0

```
44  REST_FRAMEWORK = {
45      'DEFAULT_PERMISSION_CLASSES': ('rest_framework.permissions.IsAuthenticated',),
46      'DEFAULT_RENDERER_CLASSES': (
47          'rest_framework.renderers.JSONRenderer', # default renderer
48          'rest_framework.renderers.BrowsableAPIRenderer', # default renderer
49          'rest_framework_xml.renderers.XMLRenderer',
50      ),
51      'DEFAULT_PARSER_CLASSES': (
52          'rest_framework.parsers.JSONParser', # default parser
53          'rest_framework.parsers.FormParser', # default parser
54          'rest_framework.parsers.MultiPartParser', # default parser
55          'rest_framework_xml.parsers.XMLParser',
56      ),
57  }
```

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

❖ Return List of Available Menus

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

Using Mixins
urls.py

```
16   from django.conf.urls import include, url
17   from django.contrib import admin
18   from rest_framework import routers
19   from restaurantapi.views import UserViewSet, GroupViewSet
20   from restaurantapi.views import MenuViewSet, MenuItemViewSet
21   from restaurantapi.views import AvailableMenuList
```

```
29   urlpatterns = [
30       url(r'^', include(router.urls)),
31       url(r'^admin/', include(admin.site.urls)),
32       url(r'api-auth/', include('rest_framework.urls', namespace='rest_framework')),
33       url(r'^available_menus/$', AvailableMenuList.as_view()),
34   ]
```
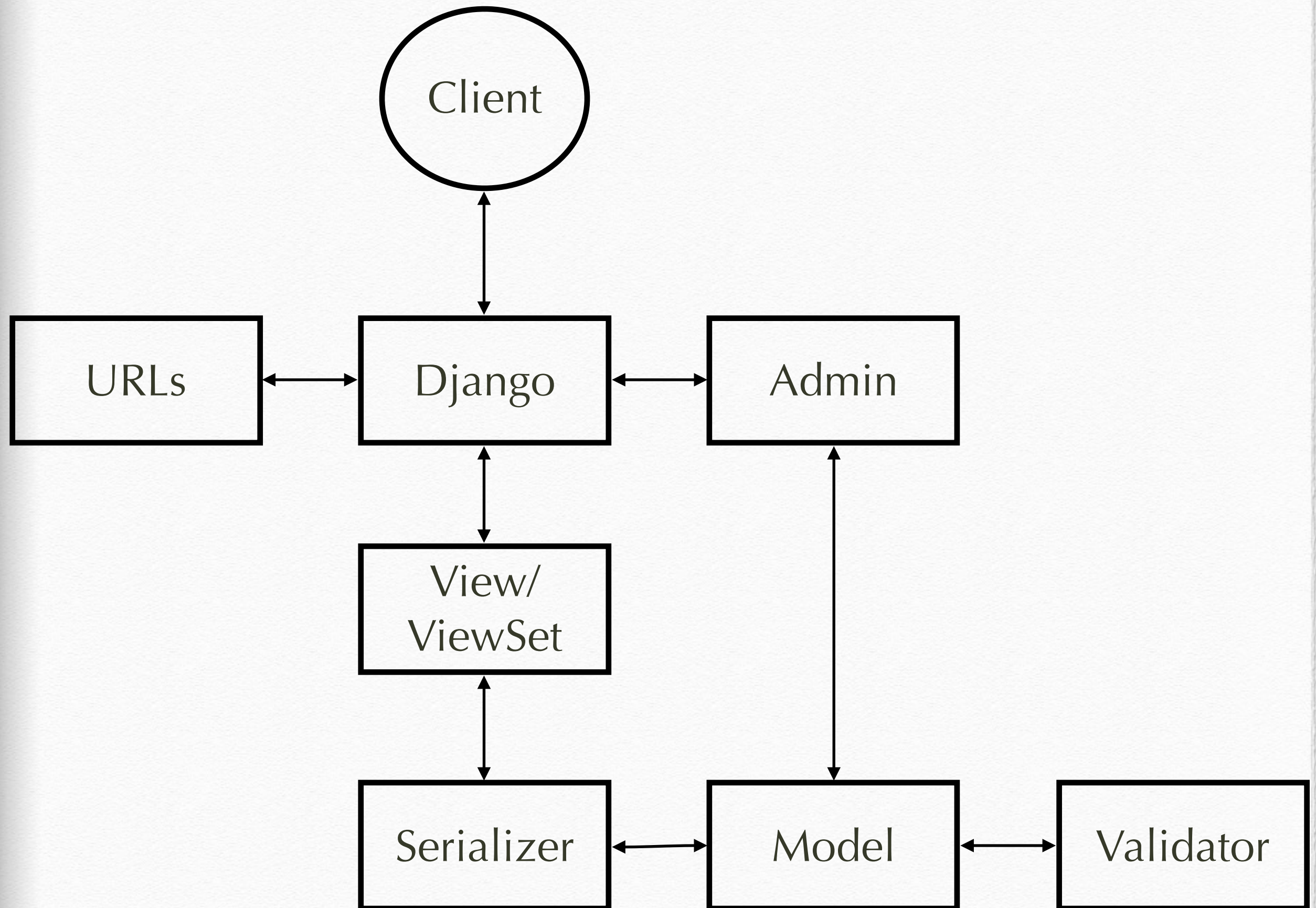
Using API View: views.py

```python
13   # for custom GET of available menu details
14   from django.http import Http404
15   from rest_framework.exceptions import APIException
16   from rest_framework.response import Response
17   from rest_framework.views import APIView
18
19   # Create your views here.
20
21   class ForbiddenAccess(APIException):
22       status_code = 403
23       default_detail = 'Action Forbidden'
24
25   class AvailableMenuDetail(APIView):
26
27       permission_classes = ()
28
29       def get_object(self, pk):
30           try:
31               return Menu.objects.get(pk=pk, available=True)
32
33           except Menu.DoesNotExist:
34               raise Http404
35
36       def get(self, request, pk, format=None):
37           menu = self.get_object(pk)
38
39           serializer = MenuSerializer(menu, context={'request':request})
40
41           return Response(serializer.data)
42
43       def put(self, request, pk, format=None):
44
45           raise ForbiddenAccess
46
47       def delete(self, request, pk, format=None):
48
49           raise ForbiddenAccess
```

71

Using API View
urls.py

```python
16    from django.conf.urls import include, url
17    from django.contrib import admin
18    from rest_framework import routers
19    from restaurantapi.views import UserViewSet, GroupViewSet
20    from restaurantapi.views import MenuViewSet, MenuItemViewSet
21    from restaurantapi.views import AvailableMenuList, AvailableMenuDetail
```

```python
29    urlpatterns = [
30        url(r'^', include(router.urls)),
31        url(r'^admin/', include(admin.site.urls)),
32        url(r'api-auth/', include('rest_framework.urls', namespace='rest_framework')),
33        url(r'^available_menus/$', AvailableMenuList.as_view()),
34        url(r'^available_menus/(?P<pk>[0-9]+)/$', AvailableMenuDetail.as_view()),
35    ]
```

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