Universitat de Lleida
Escola Politècnica Superior

Jordi Virgili
jvirgili@diei.udl.cat

PROCESS MODELS (DEGREE IN COMPUTER SCIENCE)

DJANGO TUTORIAL

Starting a project

```
django-admin.py startproject myrecommendations
cd myrecommendations
mkdir templates
```

In myrecommendations/settings.py

- Edit your database settings, for instance SQLITE:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

- And register the templates folder to use both:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates'),
os.path.join(os.path.dirname(__file__), '../templates')]
        ,
        'APP_DIRS': True,
…
```

Finally, let Django take control of the database:

```
python manage.py makemigrations
python manage.py migrate
```

Next, create a superuser (admin):

```
python manage.py createsuperuser
```

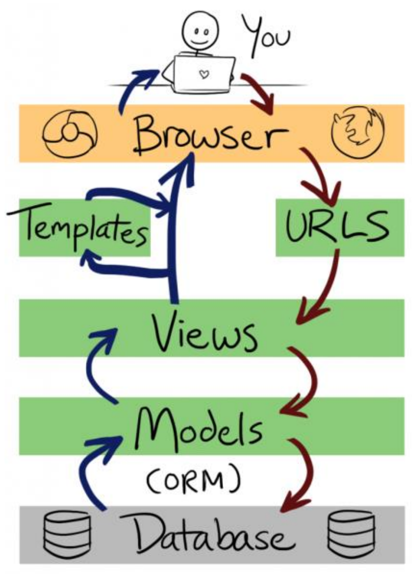
CREATING AN APPLICATION…

```
python manage.py startapp myrestaurants
```

Add 'myrestaurants', to INSTALLED_APPS list in myrecommendations/settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myrestaurants'
]
```

OVERVIEW



CREATE YOUR DATA MODEL

In myrestaurants/models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.urls import reverse
from datetime import date

class Restaurant(models.Model):
    name = models.TextField()
    street = models.TextField(blank=True, null=True)
    number = models.IntegerField(blank=True, null=True)
    city = models.TextField(default="")
    zipCode = models.TextField(blank=True, null=True)
    stateOrProvince = models.TextField(blank=True, null=True)
    country = models.TextField(blank=True, null=True)
    telephone = models.TextField(blank=True, null=True)
    url = models.URLField(blank=True, null=True)
    user = models.ForeignKey(User, default=1, on_delete=models.SET_DEFAULT)
    date = models.DateField(default=date.today)

    def __unicode__(self):
        return u"%s" % self.name
    def get_absolute_url(self):
        return reverse('myrestaurants:restaurant_detail', kwargs={'pk': self.pk})

class Dish(models.Model):
    name = models.TextField()
    description = models.TextField(blank=True, null=True)
    price = models.DecimalField('Euro amount', max_digits=8, decimal_places=2,
blank=True,
        null=True)
    user = models.ForeignKey(User, default=1, on_delete=models.SET_DEFAULT)
    date = models.DateField(default=date.today)
    restaurant = models.ForeignKey(Restaurant, null=True, related_name='dishes',
on_delete=models.PROTECT)

    def __unicode__(self):
        return u"%s" % self.name
    def get_absolute_url(self):
        return reverse('myrestaurants:dish_detail',
            kwargs={'pkr': self.restaurant.pk, 'pk': self.pk})

class Review(models.Model):
    RATING_CHOICES = ((1, 'one'), (2, 'two'), (3, 'three'), (4, 'four'), (5, 'five'))
    rating = models.PositiveSmallIntegerField('Rating (stars)', blank=False, default=3,
    choices=RATING_CHOICES)
    comment = models.TextField(blank=True, null=True)
    user = models.ForeignKey(User, default=1, on_delete=models.SET_DEFAULT)
    date = models.DateField(default=date.today)

    class Meta:
        abstract = True

class RestaurantReview(Review):
    restaurant = models.ForeignKey(Restaurant, on_delete=models.PROTECT)

    class Meta:
        unique_together = ("restaurant", "user")
```

Validate your model and commit to database:

```
python manage.py makemigrations
python manage.py migrate
```

Optionally register your model with the administrative interface (if you have the admin application enabled under INSTALLED_APPS in settings.py), so you get a CRUD UI for free in '<URL>/admin'

Note (PyCharm enables this interface when creating the project by itself)

In myrecommendations/settings.py uncomment in the list of installed applications:

```
'django.contrib.admin',
```

In myrecommendations/urls.py :

```
from django.contrib import admin

url(r'^admin/', include(admin.site.urls)),
```

or

```
urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Finally, in admin.py in the myrestaurants directory...

```
from django.contrib import admin
from myrestaurants import models

admin.site.register(models.Restaurant)
admin.site.register(models.Dish)
…
```

Design your URLs

In the project root directory, edit myrecommendations/urls.py:

```
url(r'^myrestaurants/', include(('myrestaurants.urls', 'myrestaurants'),
namespace='myrestaurants')),
```

Or

```
path('myrestaurants/', include(('myrestaurants.urls') , 'myrestaurants'),
name='myrestaurants'),
```

In the myrestaurants application directory create urls.py:

```
from django.conf.urls import url
from django.contrib.auth.decorators import login_required

from django.utils import timezone
from django.views.generic import DetailView, ListView, UpdateView
from myrestaurants.models import Restaurant, Dish
from myrestaurants.forms import RestaurantForm, DishForm
from myrestaurants.views import RestaurantCreate, DishCreate, RestaurantDetail, review

urlpatterns = [
    # List latest 5 restaurants: /myrestaurants/
    url(r'^$',
        ListView.as_view(
            queryset=Restaurant.objects.filter(date__lte=timezone.now()).order_by('-
date')[:5],
            context_object_name='latest_restaurant_list',
            template_name='myrestaurants/restaurant_list.html'),
        name='restaurant_list'),

    # Restaurant details, ex.: /myrestaurants/restaurants/1/
    url(r'^restaurants/(?P<pk>\d+)/$',
        RestaurantDetail.as_view(),
        name='restaurant_detail'),

    # Restaurant dish details, ex: /myrestaurants/restaurants/1/dishes/1/
    url(r'^restaurants/(?P<pkr>\d+)/dishes/(?P<pk>\d+)/$',
        DetailView.as_view(
            model=Dish,
            template_name='myrestaurants/dish_detail.html'),
        name='dish_detail'),

    # Create a restaurant, /myrestaurants/restaurants/create/
    url(r'^restaurants/create/$',
        RestaurantCreate.as_view(),
        name='restaurant_create'),

    # Edit restaurant details, ex.: /myrestaurants/restaurants/1/edit/
    url(r'^restaurants/(?P<pk>\d+)/edit/$',
        LoginRequiredCheckIsOwnerUpdateView.as_view(
            model=Restaurant,
            form_class=RestaurantForm),
        name='restaurant_edit'),



    # Create a restaurant dish, ex.: /myrestaurants/restaurants/1/dishes/create/
    url(r'^restaurants/(?P<pk>\d+)/dishes/create/$',
        DishCreate.as_view(),
        name='dish_create'),


    # Create a restaurant review, ex.: /myrestaurants/restaurants/1/reviews/create/
    url(r'^restaurants/(?P<pk>\d+)/reviews/create/$',
        review,
        name='review_create'),
]
```

Custom Class Views

In myrestaurants/views.py:

```
from django.urls import reverse
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404
from django.views.generic import DetailView
from django.views.generic.edit import CreateView
from myrestaurants.models import RestaurantReview, Restaurant, Dish
from myrestaurants.forms import RestaurantForm, DishForm

class LoginRequiredMixin(object):
    @method_decorator(login_required())
    def dispatch(self, *args, **kwargs):
        return super(LoginRequiredMixin, self).dispatch(*args, **kwargs)

class CheckIsOwnerMixin(object):
    def get_object(self, *args, **kwargs):
        obj = super(CheckIsOwnerMixin, self).get_object(*args, **kwargs)
        if not obj.user == self.request.user:
            raise PermissionDenied
        return obj

class LoginRequiredCheckIsOwnerUpdateView(LoginRequiredMixin, CheckIsOwnerMixin,
UpdateView):
    template_name = 'myrestaurants/form.html'


class RestaurantDetail(DetailView):
    model = Restaurant
    template_name = 'myrestaurants/restaurant_detail.html'

    def get_context_data(self, **kwargs):
        context = super(RestaurantDetail, self).get_context_data(**kwargs)
        context['RATING_CHOICES'] = RestaurantReview.RATING_CHOICES
        return context

class RestaurantCreate(CreateView):
    model = Restaurant
    template_name = 'myrestaurants/form.html'
    form_class = RestaurantForm

    def form_valid(self, form):
        form.instance.user = self.request.user
        return super(RestaurantCreate, self).form_valid(form)

class DishCreate(CreateView):
    model = Dish
    template_name = 'myrestaurants/form.html'
    form_class = DishForm

    def form_valid(self, form):
        form.instance.user = self.request.user
        form.instance.restaurant = Restaurant.objects.get(id=self.kwargs['pk'])
        return super(DishCreate, self).form_valid(form)

def review(request, pk):
    restaurant = get_object_or_404(Restaurant, pk=pk)
    review = RestaurantReview(
        rating=request.POST['rating'],
        comment=request.POST['comment'],
        user=request.user,
        restaurant=restaurant)
    review.save()
    return HttpResponseRedirect(reverse('myrestaurants:restaurant_detail',
        args=(restaurant.id,)))
```

Create your Application Templates

First, create a base template in myrestaurants/templates/myrestaurants/base.html

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="{% static "style/base.css" %}" />
    <title>{% block title %}MyRestaurants by MyRecommentdations.org{% endblock %}</title>
</head>
<body>
<div id="header">
    {% block header %}
    {% endblock %}
</div>
<div id="sidebar">
    {% block sidebar %}<ul><li><a href="/myrestaurants">Home</a></li></ul>{% endblock %}
</div>
<div id="content">
    {% block content %}
        {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
    {% endblock %}
</div>
<div id="footer">{% block footer %}{% endblock %}</div>
</body>
</html>
```

Next create *restaurant_list.html* in *myrestaurants/templates/myrestaurants*

```
{% extends "myrestaurants/base.html" %}
{% block content %}
<h1>
    Restaurants
    {% if user %}(<a href="{% url 'myrestaurants:restaurant_create' %}">add</a>){% endif %}
</h1>
<ul>
    {% for restaurant in latest_restaurant_list %}
        <li><a href="{% url 'myrestaurants:restaurant_detail' restaurant.id %}">
            {{ restaurant.name }}</a></li>
    {% empty %}<li>Sorry, no restaurants registered yet.</li>
    {% endfor %}
```

```
</ul>
{% endblock %}
```

And *restaurant_detail.html*, which includes the list of dishes and the review form:

```
{% extends "myrestaurants/base.html" %}
{% block content %}
<h1>
    {{ restaurant.name }}
    {% if user == restaurant.user %}
        (<a href="{% url 'myrestaurants:restaurant_edit' restaurant.id %}">edit</a>)
    {% endif %}
</h1>
<h2>Address:</h2>
<p>
    {{ restaurant.street }}, {{ restaurant.number }} <br/>
    {{ restaurant.zipcode }} {{ restaurant.city }} <br/>
    {{ restaurant.stateOrProvince }} ({{ restaurant.country }})
</p>


<h2>
    Dishes
    {% if user %}
        (<a href="{% url 'myrestaurants:dish_create' restaurant.id %}">add</a>)
    {% endif %}
</h2>
<ul>
    {% for dish in restaurant.dishes.all %}
        <li><a href="{% url 'myrestaurants:dish_detail' restaurant.id dish.id %}">
            {{ dish.name }}</a></li>
    {% empty %}<li>Sorry, no dishes for this restaurant yet.</li>
    {% endfor %}
</ul>
<h2>Reviews</h2>
<ul>
    {% for review in restaurant.restaurantreview_set.all %}
        <li>
            <p>{{ review.rating }} star{{ review.rating|pluralize }}</p>
            <p>{{ review.comment }}</p>
            <p>Created by {{ review.user }} on {{ review.date }}</p>
        </li>
    {% endfor %}
</ul>
<h3>Add Review</h3>
<form action="{% url 'myrestaurants:review_create' restaurant.id %}" method="post">
    {% csrf_token %}
    Message: <textarea name="comment" id="comment" rows="4"></textarea>
    <p>Rating:</p>
    <p>{% for rate in RATING_CHOICES %}
    <input type="radio" name="rating" id="rating{{ forloop.counter }}" value="{{ rate.1 }}" />
    <label for="choice{{ forloop.counter }}">{{ rate.1 }} star{{ rate.0|pluralize }}</label>
    <br/>{% endfor %}
    </p>
    <input type="submit" value="Review" />
</form>

{% endblock %}
{% block footer %}
    Created by {{ restaurant.user }} on {{ restaurant.date }}
{% endblock %}
```

CREATE FORMS

Finally, there are the forms in *myrestaurant/forms.py* that are automatically created from the Restaurant and Dish models to create and edit them:

```
from django.forms import ModelForm
from .models import Restaurant, Dish

class RestaurantForm(ModelForm):
    class Meta:
        model = Restaurant
        exclude = ('user', 'date',)
```

```
class DishForm(ModelForm):
    class Meta:
        model = Dish
        exclude = ('user', 'date', 'restaurant',)
```

And the template that shows them, *form.html*:

```
{% extends "myrestaurants/base.html" %}
{% block content %}
    <form method="post" action="">
        {% csrf_token %}
        <table>
            {{ form.as_table }}
        </table>
        <input type="submit" value="Submit"/>
    </form>
{% endblock %}
```

Finally, update database:

```
python manage.py makemigrations
python manage.py migrate
```