

# Global Startup

## Tech Meet-up 2: Models, Views, and Templates



# Today's Meet-up

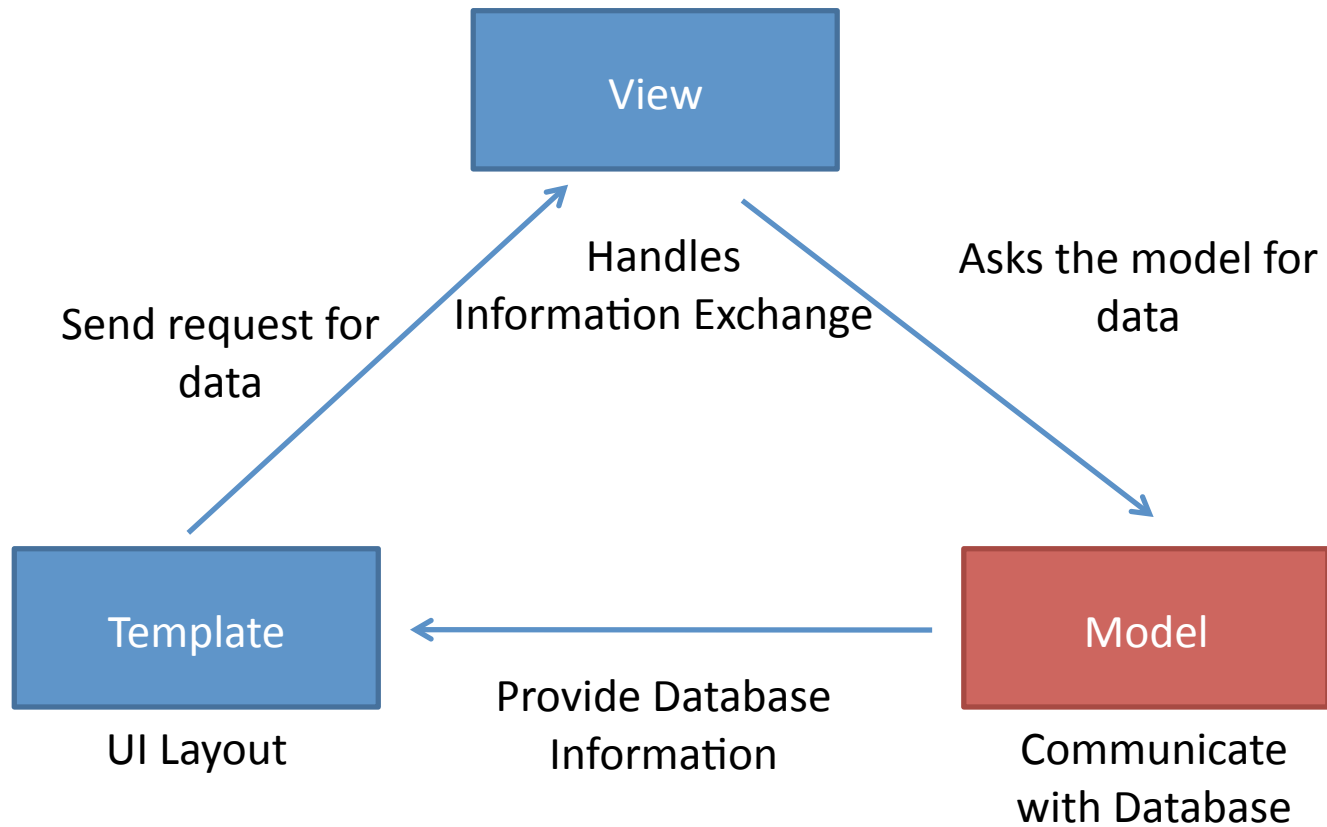
- Back to the big picture
- Review of Models
- Templates
- Views
- Today's Assignment

# The Big Picture



# Models

# Models



# What is a Model

- Python class describing data in your application
  - Subclass of `models.Model`
- Assigns attributes to each data field
- Avoid direct work with the database
  - No need to handle database connections, timeouts, etc. – Let Django do it for you!
  - Provides Schema for database

# Django Model Syntax

Import statements

SubClass of  
models.Model Class

Define fields

`__unicode__`  
corresponds to  
python `__str__`

Can define more  
functions

```
from django.db import models
import datetime
from django.utils import timezone

# Create your models here.

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

    def __unicode__(self):
        return self.question

    def was_published_recently(self):
        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)
```

# Django Fields

We can define fields directly in our model class

- No need to define manually in database

Example: create two fields in our Poll class

```
class Poll(models.Model):  
    question = models.CharField(max_length=200)  
    pub_date = models.DateTimeField('date published')
```

Define Type of Field

- E.g. `models.CharField`

Define arguments of field

- E.g. `max_length=200`

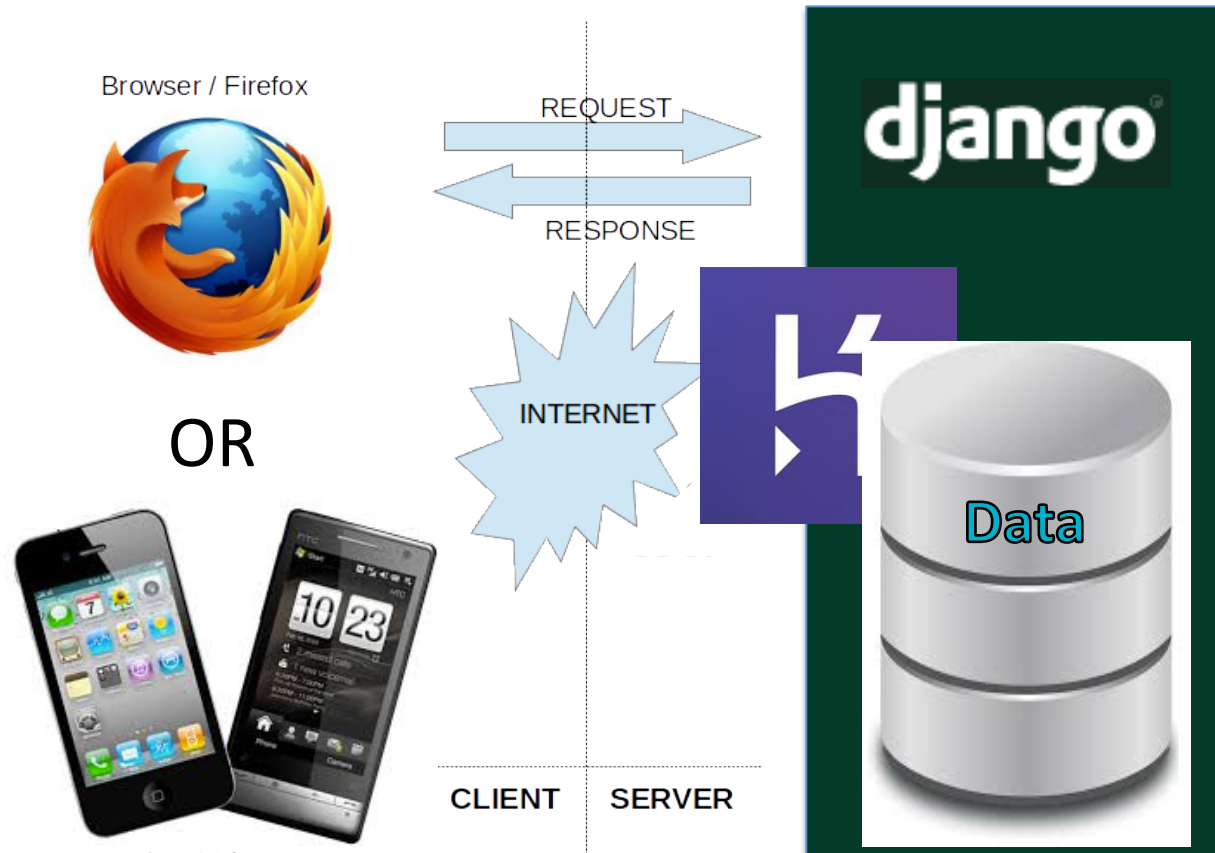
Django will automatically create fields in database





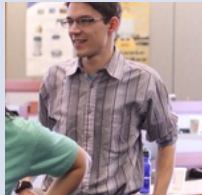
# Important Django Field Types

- BooleanField
  - Checkbox
- CharField(max\_length)
  - Single-line textbox
- DateField
  - Javascript calendar
- DateTimeField
  - Javascript calendar, time picker
- DecimalField(max\_digits, decimal\_places)
  - Decimal numbers
- EmailField
  - Charfield that validates email address
- FileField
  - File upload, stores path in database
- FloatField
  - Floating point numbers
- IntegerField
  - Integer textbox
- PositiveIntegerField
  - Integer textbos for positive integers
- TextField
  - Multi-line textbox

# The Big Picture



# Databases

ID	First name	Email	Profile photo	...
1	Kayla	kmeduna@alum.mit.edu		
2	Wesley	wgraybill@alum.mit.edu		
3	Cole	houstonc@mit.edu		

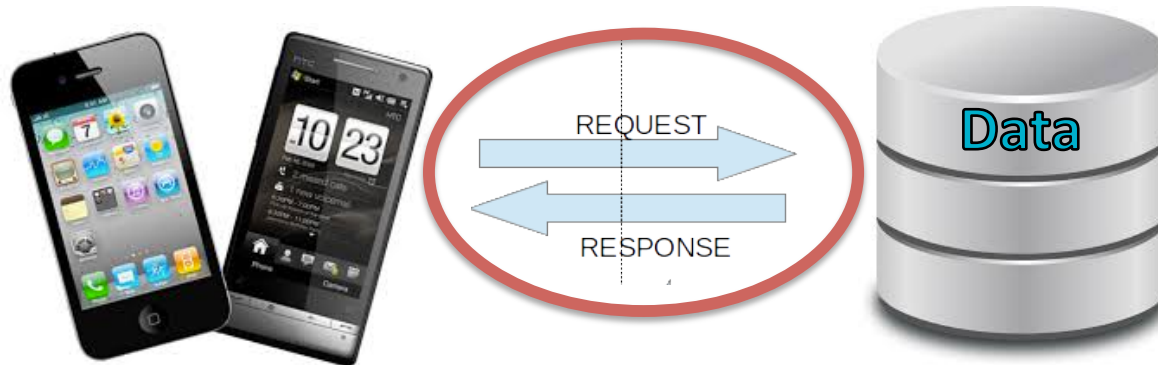
# Databases

ID	First	Email	Profile	
	ID	User ID	Post Text	
1	1	1	Wesley is the best Tech instructor ever!!! I'm so glad I'm teaching with him!	
2				
3	2	3	I love that all my students are so talkative and ask questions in class	

# The Big Picture



# The Big Picture



# RESTful API

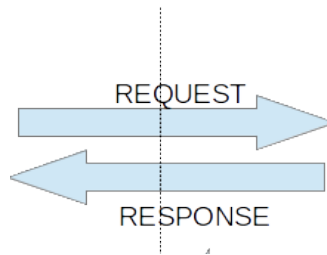


# Example: Facebook API

Retrieve a user's posts

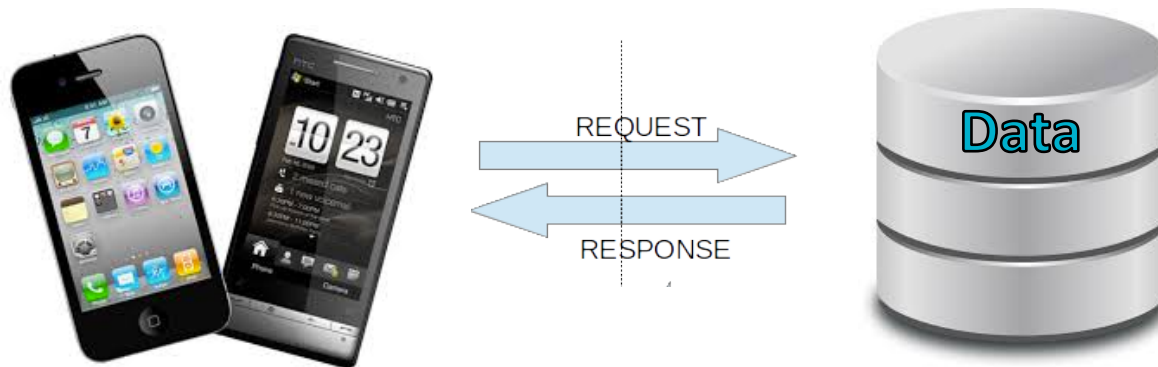


```
GET graph.facebook.com  
/<user_id>/posts
```




## Facebook queries database:

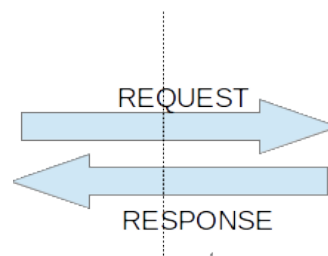
```
user = User.objects.get(id = 1)
posts = user.post_set.all()
```



## Facebook responds with posts JSON:



```
[{ post: { id: 1, text: 'Wesley is  
the best...' }},  
{ post: { id: 5, text: 'The students  
who bribe me with the best Malaysian  
food get As' }}]
```



# REST verbs

**GET**

Retrieve data

**POST**

Create new data

**PUT**

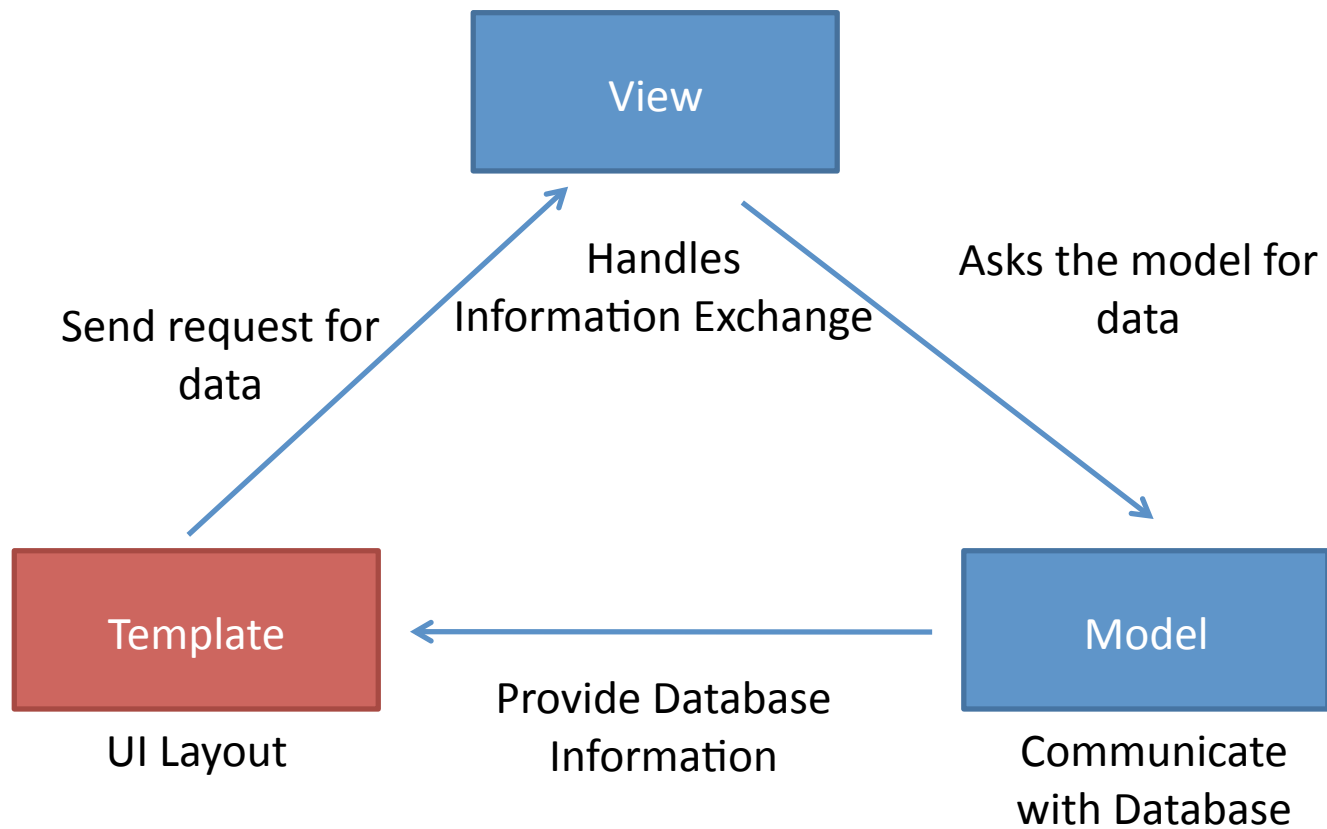
Update existing data

**DELETE**

Delete data

# Templates

# Templates



# What is a template?

- A text-based template for HTML, CSS, XML, JavaScript, etc.
- Defines the Layout of your Webpage in [name].html in your template directory
- Can contain
  - Hard coded text
  - Variables: `{ { ... } }`
  - Tags: `{ % ... % }`

<https://docs.djangoproject.com/en/1.6/topics/templates/>

# Django Template Syntax

```
<h2>The Latest 5 Polls</h2>
{% if latest_poll_list %}
  <ul>
    {% for poll in latest_poll_list %}
      <li><a href="/polls/{{ poll.id }}">{{ poll.question }}</a></li>
    {% endfor %}
  </ul>
{% else %}
  <p>No polls are available.</p>
{% endif %}
```

## The Latest 5 Polls

- [What's new?](#)

Hard coded text

Tags: {% ... %}  
that control logic

Variables: {{ ... }}  
that get replaced



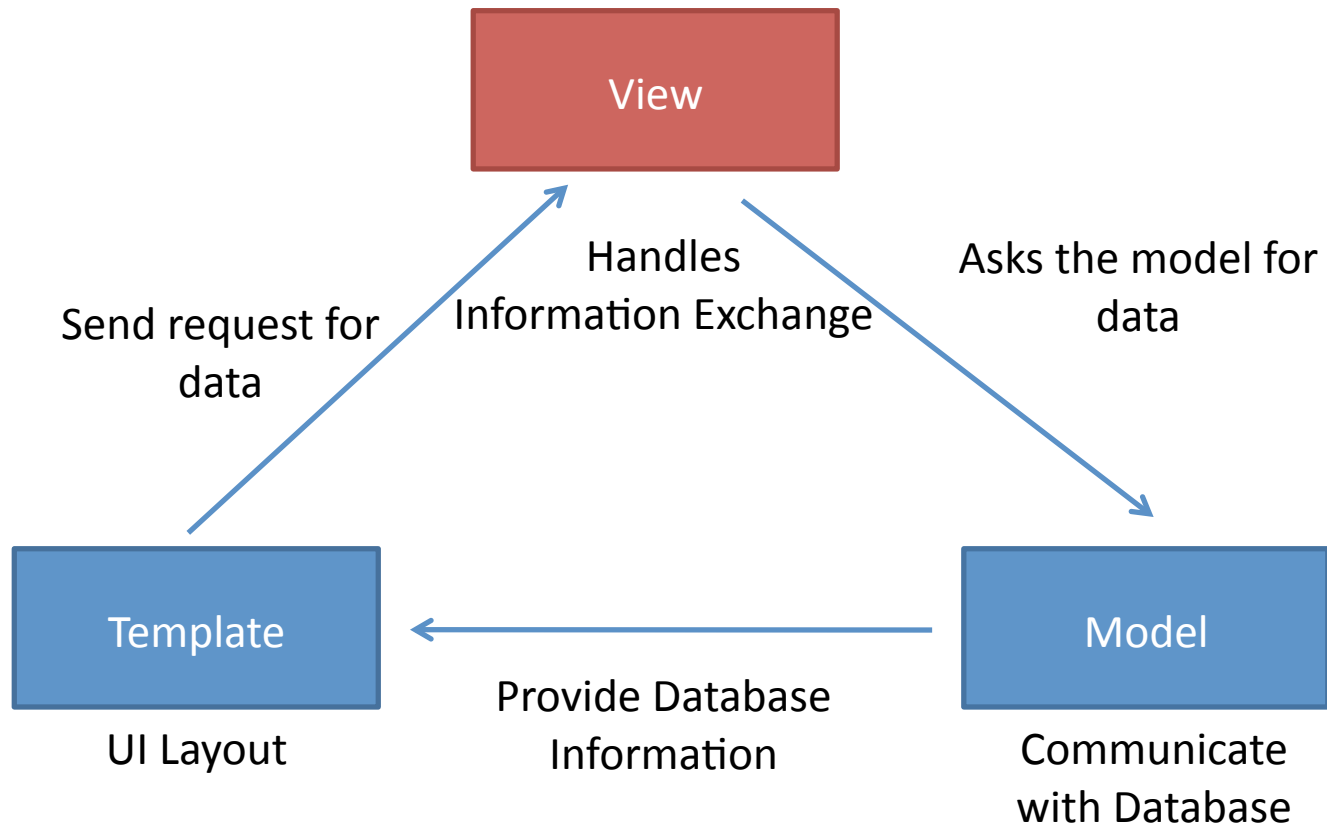
# Setting Variables

Templates by themselves are not very useful

How do we insert the content from our database into our UI?

Use **Views** to connect **Models** and **Templates!**

# Views

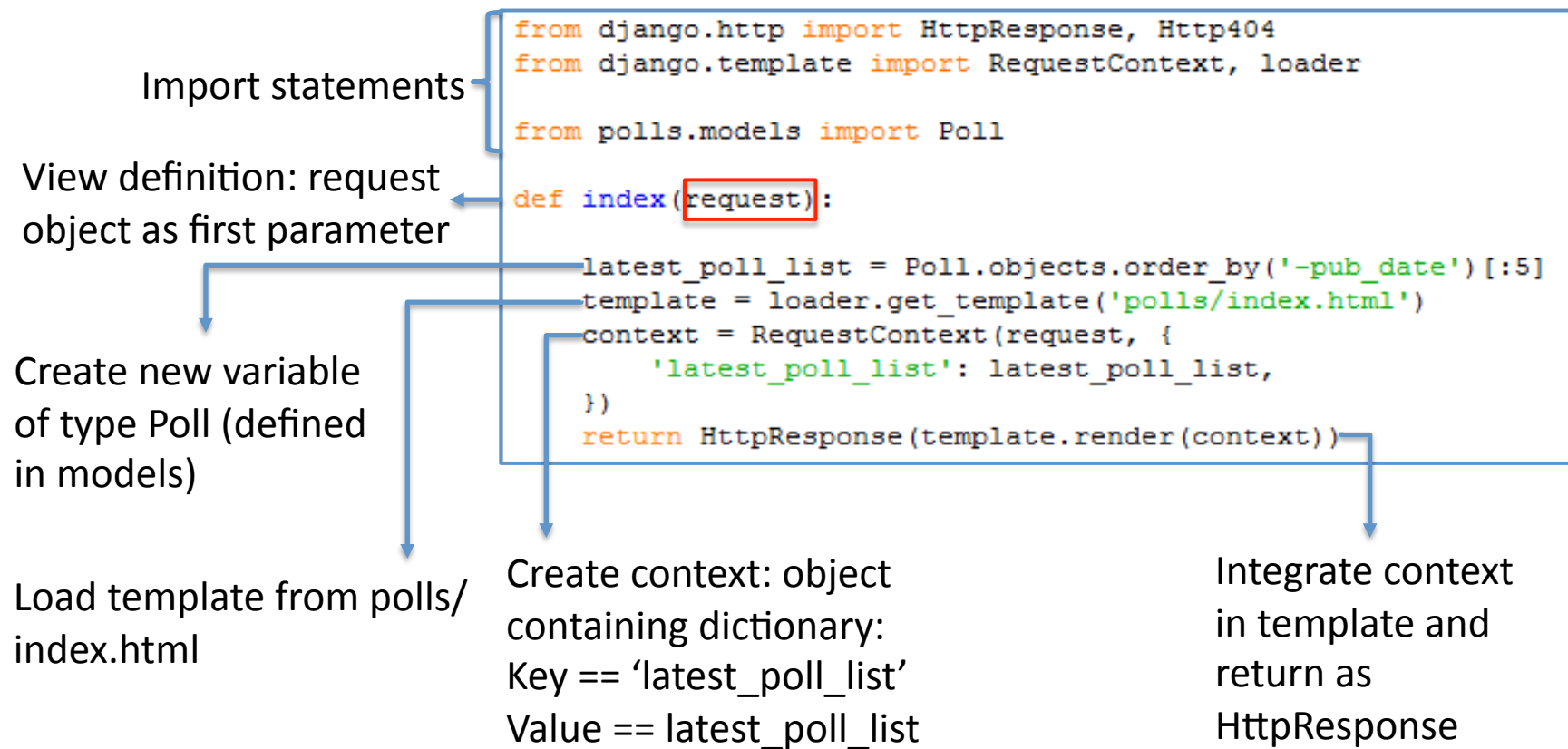


# What is a view

- Views are the logical interface between data (Models) and presentation (Templates)
- Defined in **views.py** inside the *appFolder*
- EVERY view takes a request object as first parameter
- EVERY view returns an **HttpResponse** object

```
from django.http import HttpResponse
def hello(request):
    return HttpResponse("Hello world")
```

# Django View Syntax



# Directing http requests to views

e.g. user requests `http://[host]/polls/12/results`

1.: look in `[project]/mysite/urls.py`  
(set in `settings.py`)

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^admin/', include(admin.site.urls)),
    url(r'^polls/', include('polls.urls')),
)
```

2.: Direct to `[project]/mysite/polls/urls.py`

```
from django.conf.urls import patterns, url

from polls import views

urlpatterns = patterns('',
    # ex: /polls/
    url(r'^$', views.index, name='index'),
    # ex: /polls/5/
    url(r'^(?P<poll_id>\d+)/$', views.detail, name='detail'),
    # ex: /polls/5/results/
    url(r'^(?P<poll_id>\d+)/results/$', views.results, name='results'),
    # ex: /polls/5/vote/
    url(r'^(?P<poll_id>\d+)/vote/$', views.vote, name='vote'),
)
```

3.: Find url and load  
`views.results`

# Request Life Cycle

1. User requests to view URL
2. Django determines the root URLconf by looking at the `ROOT_URLCONF` setting.
3. Django looks at all of the URLpatterns in the URLconf for the *first* one that matches the URL
4. If it finds a match, it calls the associated view function.
  - Repeats Steps 3-4 if redirected to another `urls.py`
5. The view function returns an `HttpResponse`.
6. Django converts the `HttpResponse` to the proper HTTP response, which results in a Web page.

# Today's Assignment

# Today's Assignment

## Carry on with the Django Tutorial

- Create an Admin page for your Website

[Writing your first Django App – Part 2](#)

- Learn how to configure Views and Templates

[Writing your first Django App – Part 3](#)

## Helpful Documentation:

- [View](#) and [Template](#) section of Django docs