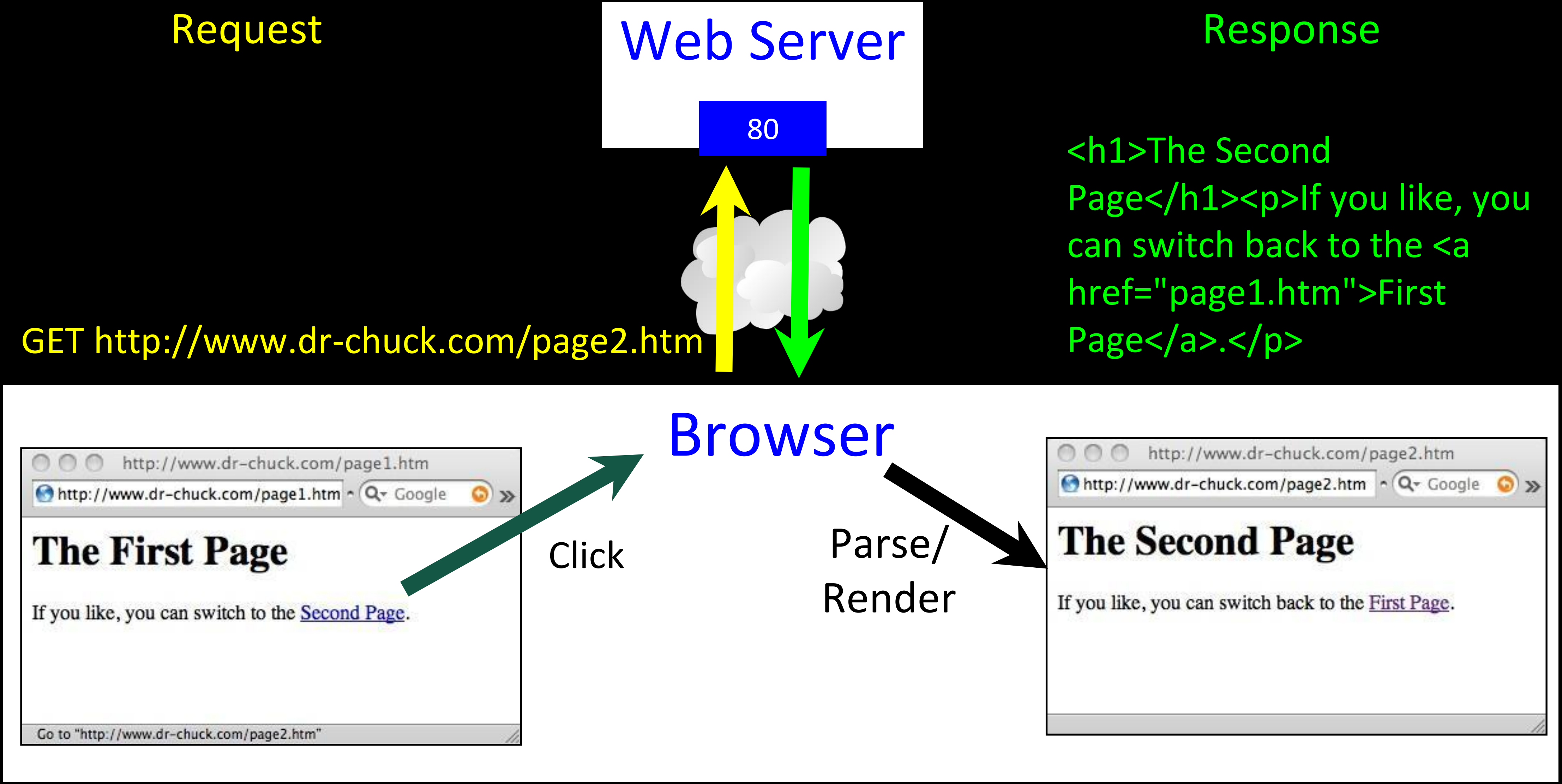
Introdução a Django

Ricardo Costa

rac2@cesar.school

Model View Controller (MVC)

Charles Severance
www.dj4e.com


Request
Web Server
80
Response
GET http://www.dr-chuck.com/page2.htm
Browser
Click
Parse/
Render
http://www.dr-chuck.com/page1.htm
Google
Go to "http://www.dr-chuck.com/page2.htm"
http://www.dr-chuck.com/page2.htm
Google

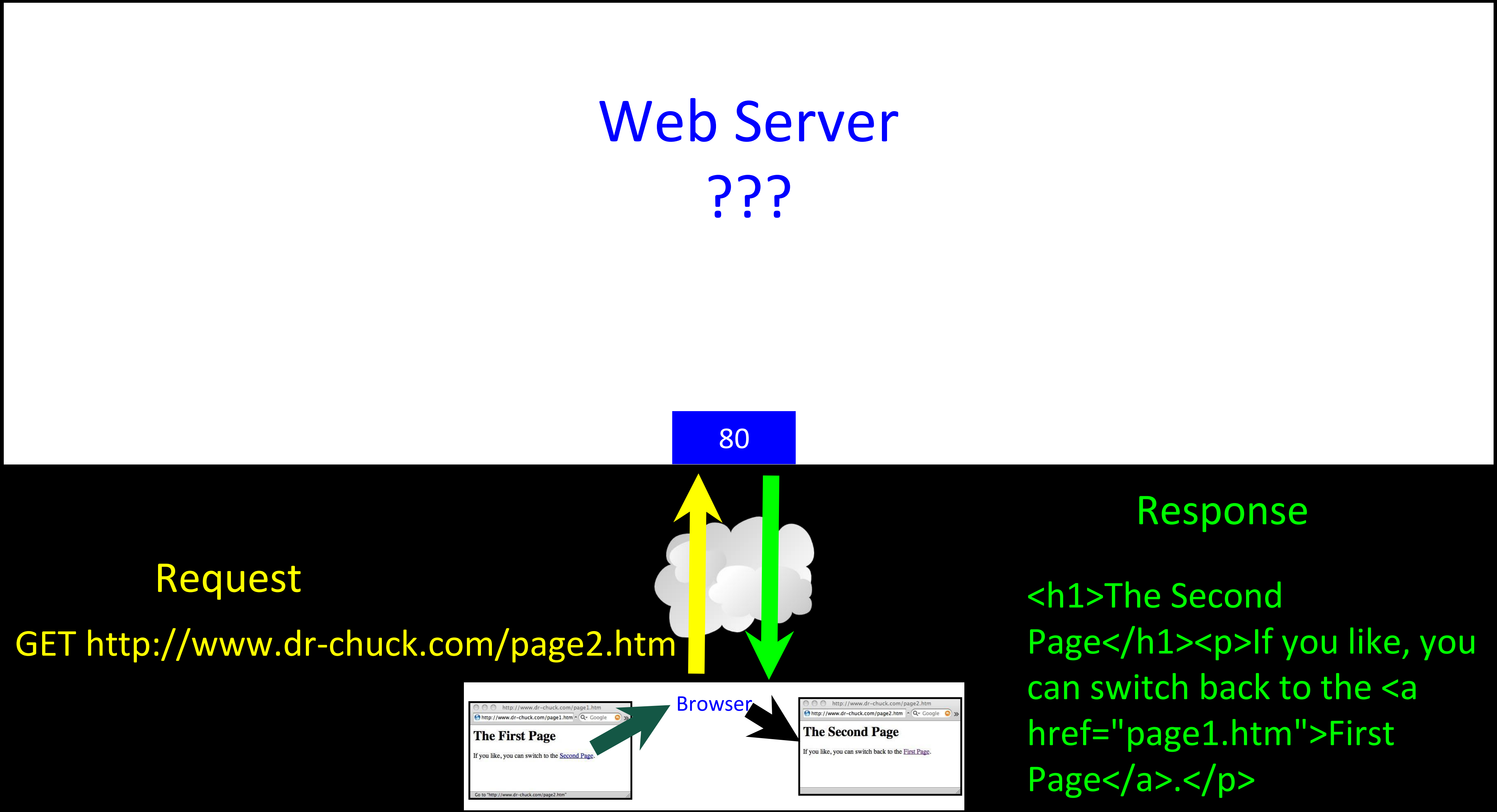
```
<h1>The Second Page</h1><p>If you like, you can switch back to the <a href="page1.htm">First Page</a>.</p>
```

The First Page

If you like, you can switch to the Second Page.

The Second Page

If you like, you can switch back to the First Page.



http://www.dr-chuck.com/page1.htm

The First Page

If you like, you can switch to the Second Page.

http://www.dr-chuck.com/page2.htm

The Second Page

If you like, you can switch back to the First Page.

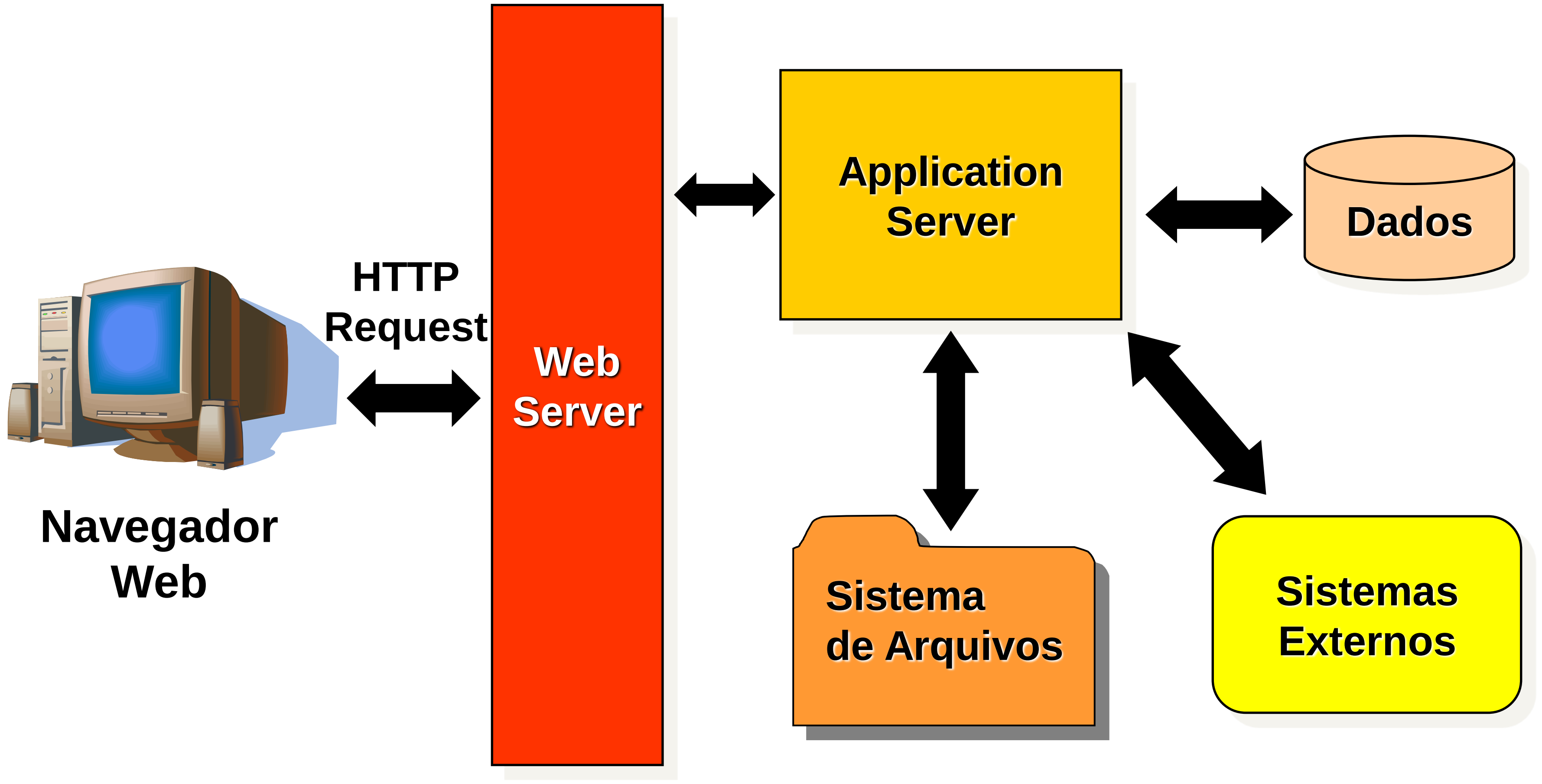
Request

GET http://www.dr-chuck.com/page2.htm

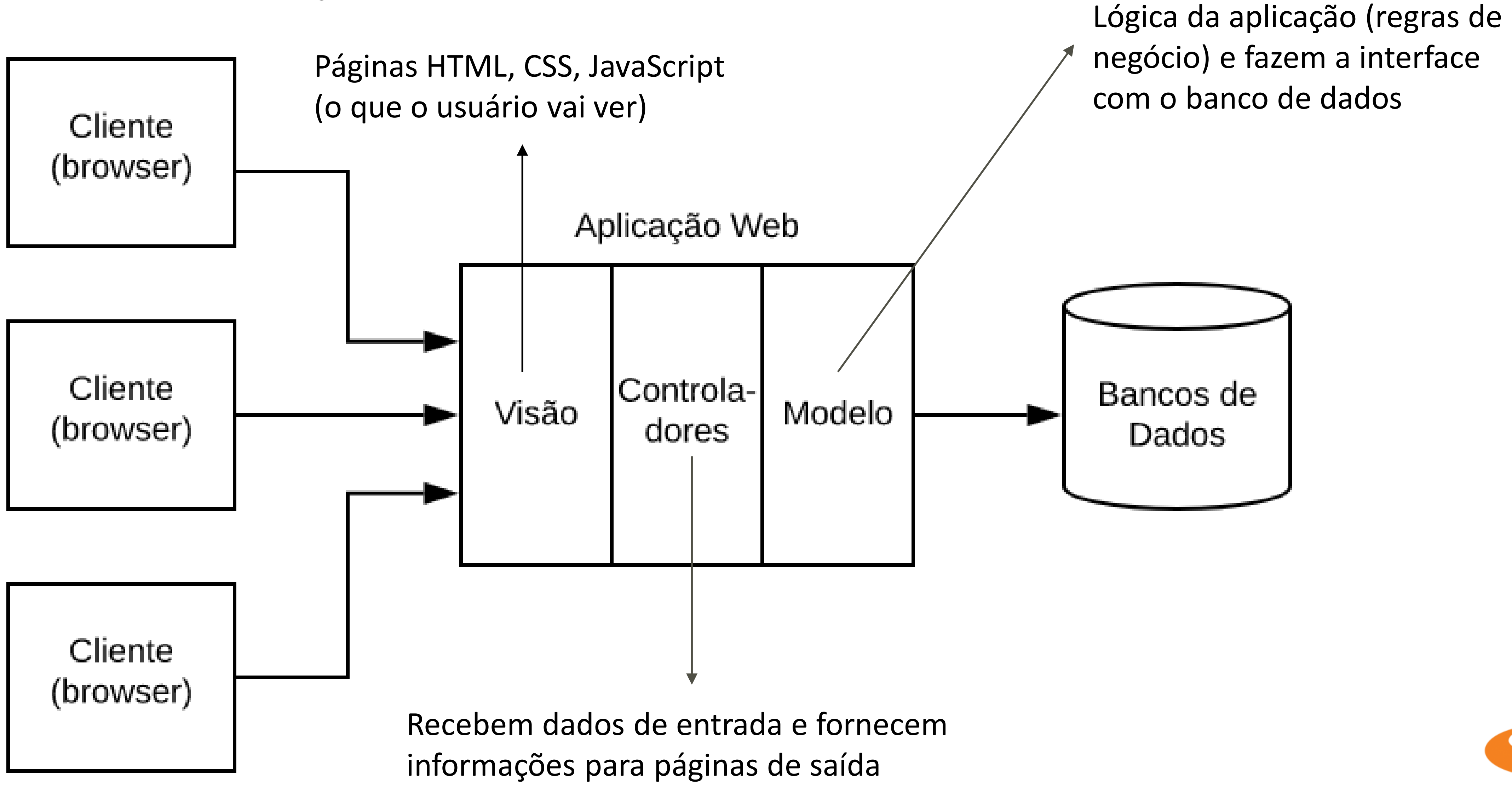
Response

```
<h1>The Second
Page</h1><p>If you like, you
can switch back to the <a
href="page1.htm">First
Page</a>.</p>
```

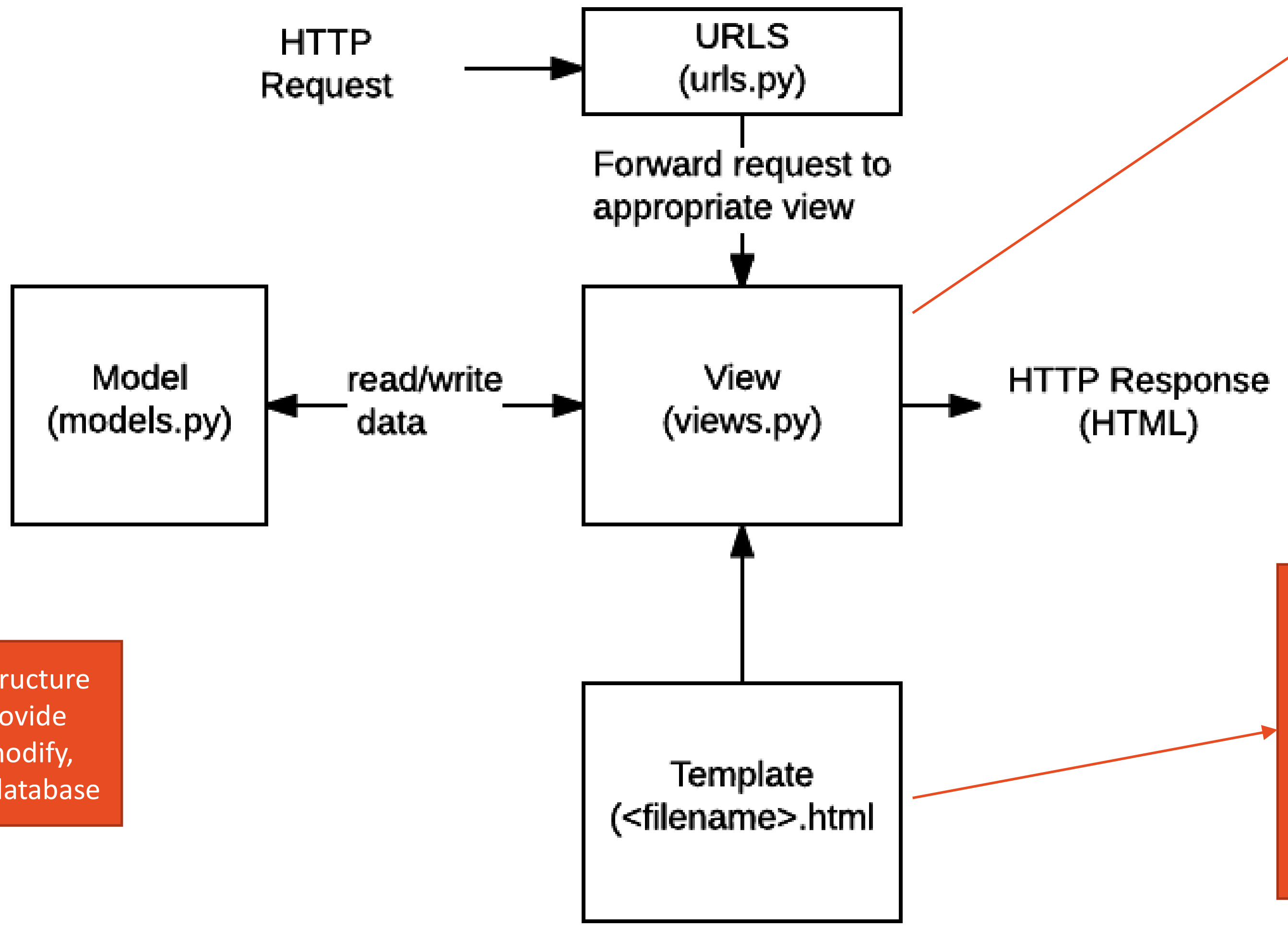
Típica arquitetura de uma Aplicação Web



Padrão Arquitetural MVC


Páginas HTML, CSS, JavaScript
(o que o usuário vai ver)
Lógica da aplicação (regras de negócio) e fazem a interface com o banco de dados
Cliente (browser)
Cliente (browser)
Cliente (browser)
Aplicação Web
Visão
Controla-dores
Modelo
Bancos de Dados
Recebem dados de entrada e fornecem informações para páginas de saída

Django e o padrão MVT



A request handler function. Receives HTTP requests and returns HTTP responses. Access the data needed via models, and delegate the formatting of the response to templates.

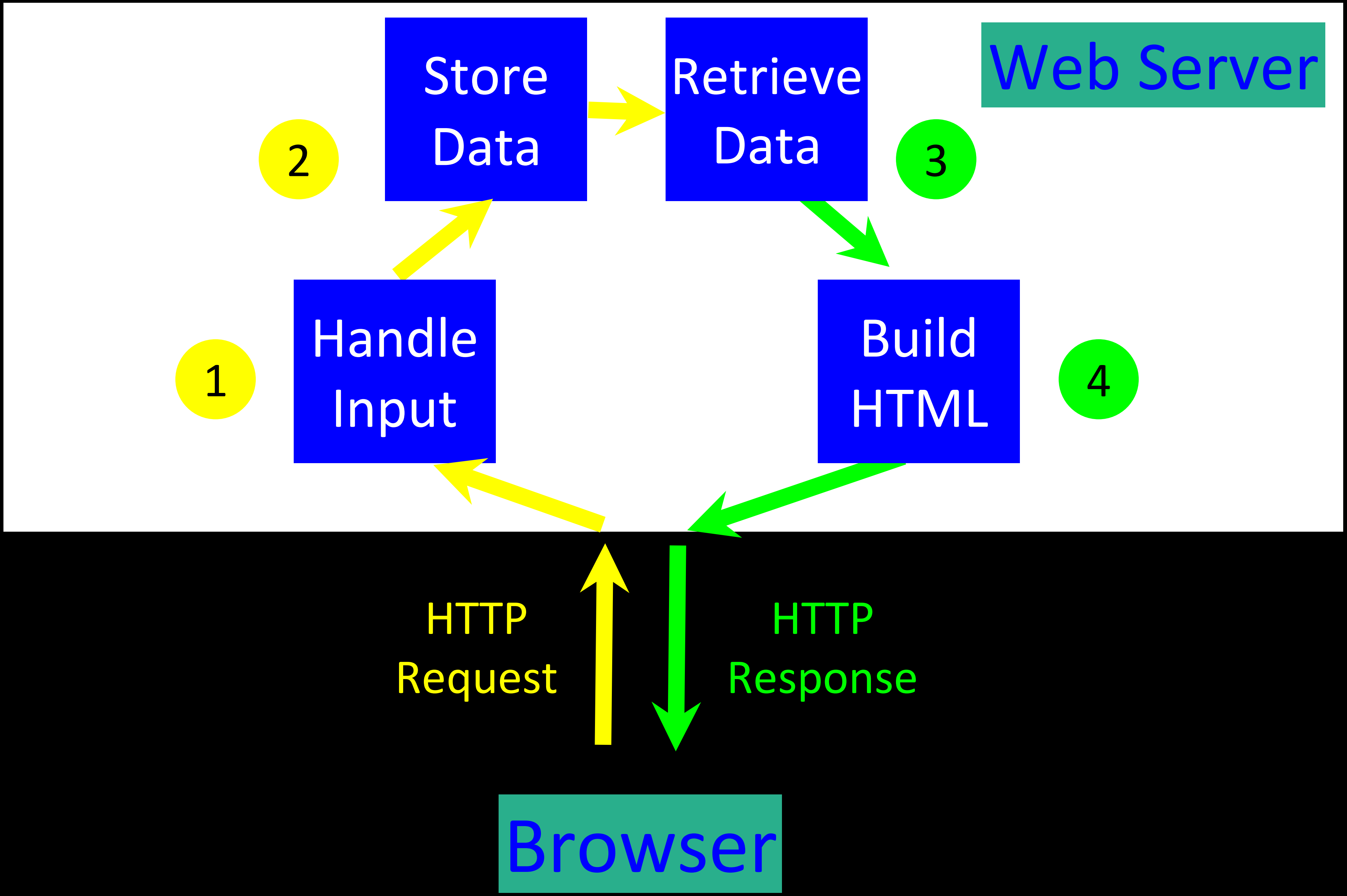
Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database

A text file defining the structure of a file (such as an HTML page), with placeholders used to represent actual content.
A view can dynamically create an HTML page using an HTML template, populating it with data from a model.

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

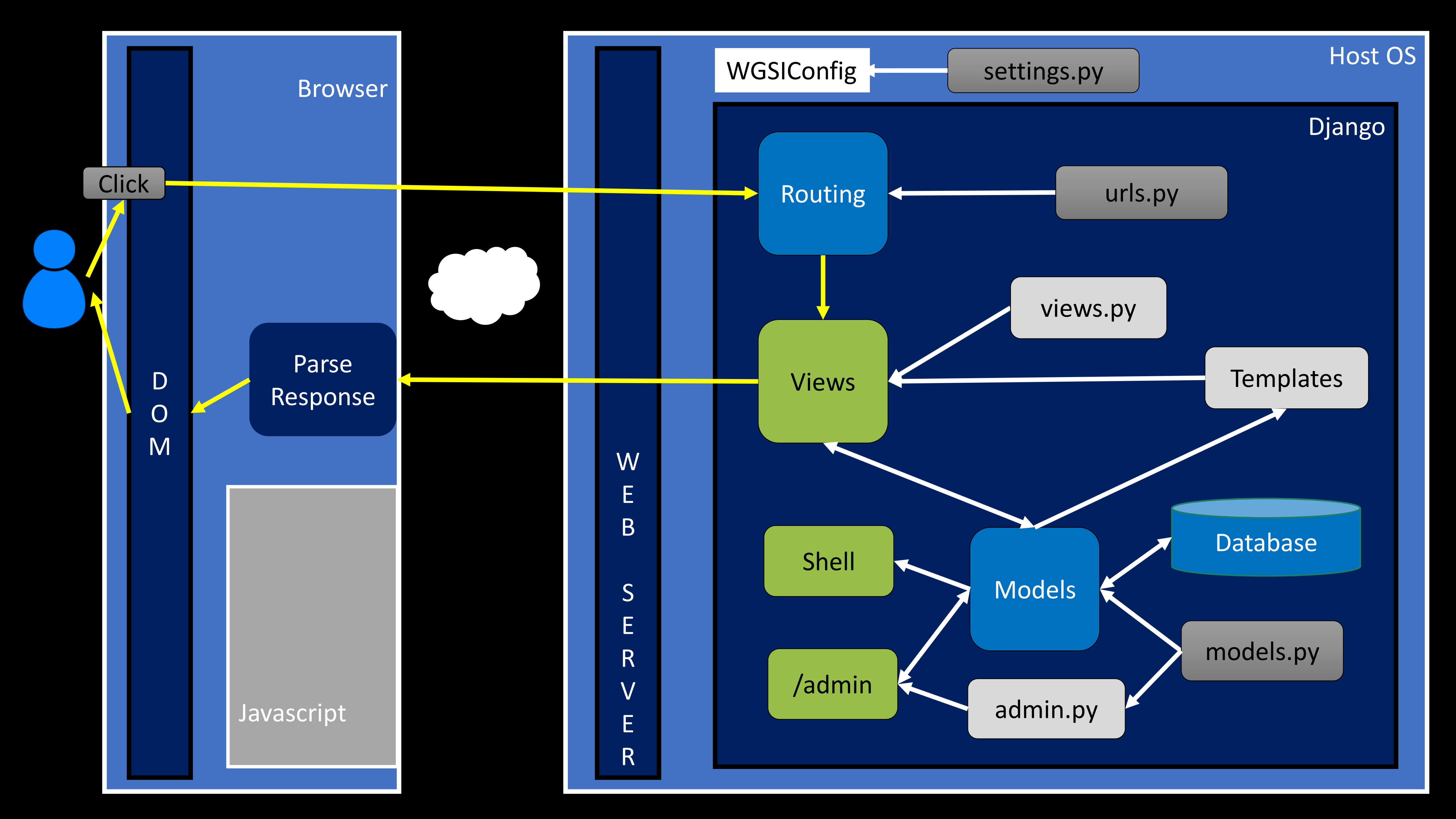
Tasks Inside the Server

- Process any user input data (i.e. from a form) - possibly storing it in a database or making some other change to the database such as a delete
- Decide which screen to send back to the user
- Retrieve any needed data
- Produce the HTML response and send it back to the browser (i.e. a template)

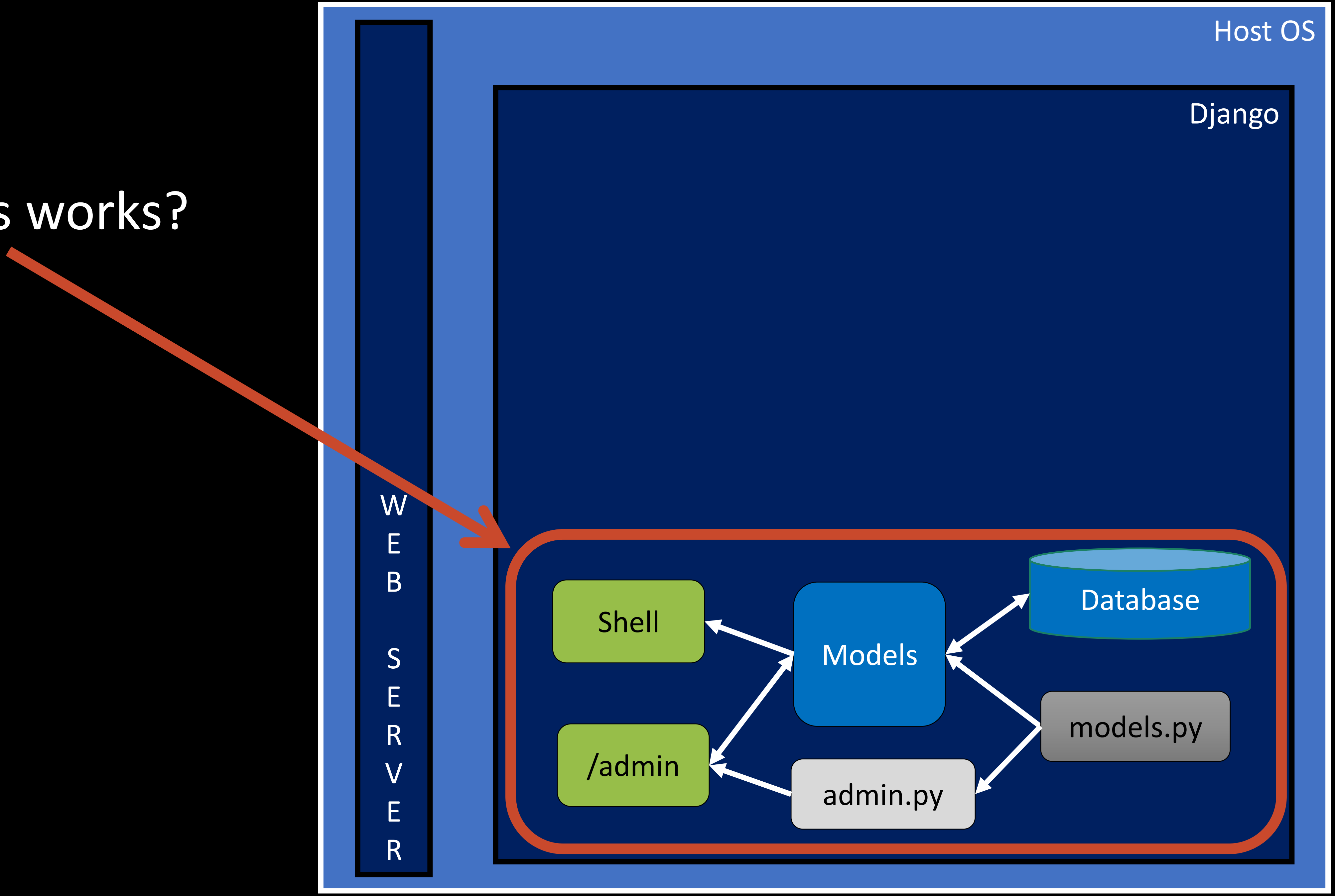

Web Server
Store Data
Retrieve Data
2
3
1
Handle Input
Build HTML
4
HTTP Request
HTTP Response
Browser

Prática 01

- Vamos configurar nosso ambiente e criar o projeto Django
- Nessas aulas criaremos um “*super simple stackoverflow*”


Browser
Click
D O M
Parse Response
Javascript
W E B S E R V E R
Host OS
WGSIConfig
settings.py
Django
Routing
urls.py
views.py
Views
Templates
Shell
Models
Database
/admin
models.py
admin.py

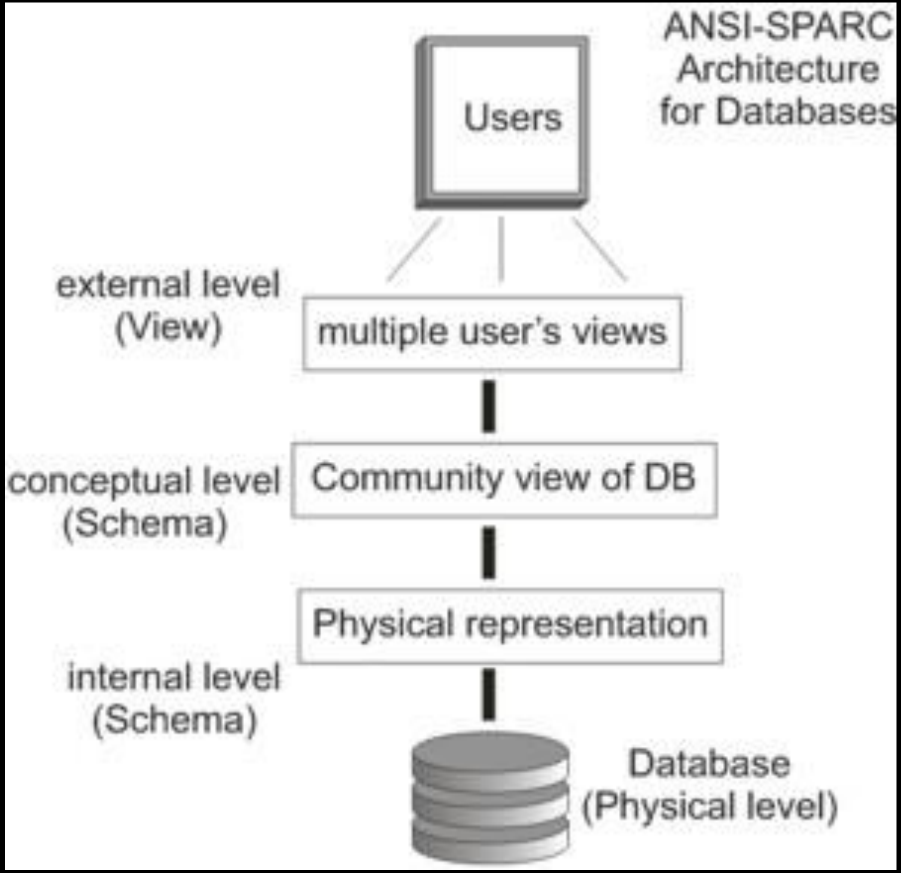
How this works?



SQL

Structured Query Language is the language we use to issue commands to the database

- Create/Insert data
- Read/Select some data
- Update data
- Delete data



http://en.wikipedia.org/wiki/SQL
https://en.wikipedia.org/wiki/ANSI-SPARC_Architecture

Start Simple - A Single Table

```
CREATE TABLE Users(
    id integer NOT NULL
      PRIMARY KEY
      AUTOINCREMENT,
  name VARCHAR(128),
  email VARCHAR(128)
);
```

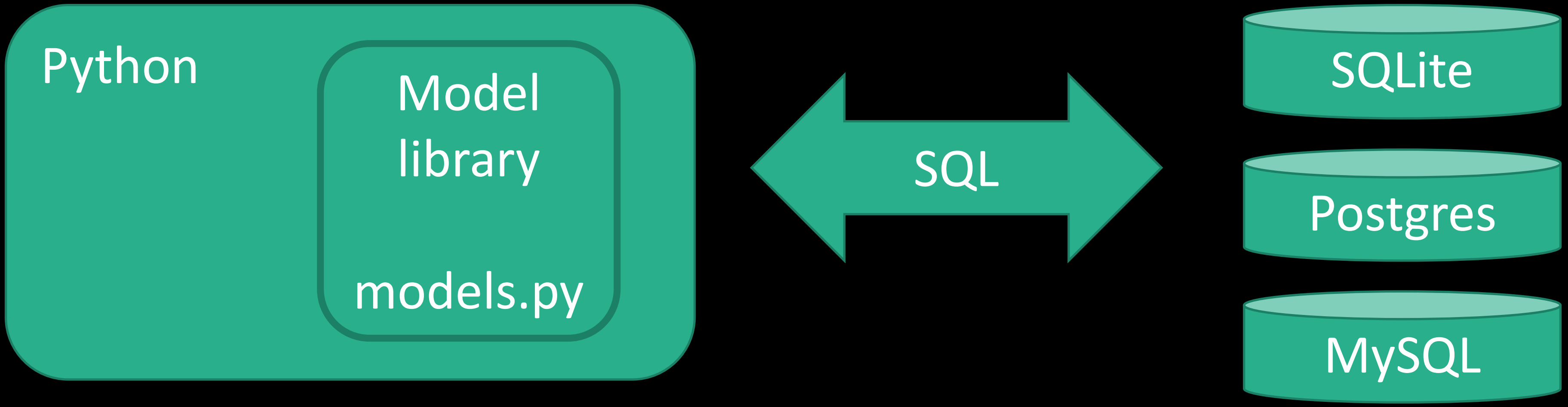
https://www.dj4e.com/lectures/SQL-01-Basics.txt

SQL Summary

```
INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')

DELETE FROM Users WHERE email='ted@umich.edu'

UPDATE Users SET name="Charles" WHERE email='csev@umich.edu'

SELECT * FROM Users

SELECT * FROM Users WHERE email='csev@umich.edu'

SELECT * FROM Users ORDER BY email
```

Object Relational Mapping (ORM)

- Allows us to map tables to objects and columns
- We use those objects to store and retrieve data from the database
- Improved portability across database dialects (SQLite, MySQL, Postgres, Oracle)



Defining a table

SQL:

```
CREATE TABLE Users(
  name VARCHAR(128),
  email VARCHAR(128)
);
```

models.py:

```
from django.db import models

class User(models.Model):
    name = models.CharField(max_length=128)
    email = models.CharField(max_length=128)
```

https://github.com/csev/dj4e-samples/tree/master/users

Creating the Table from the Model

```
$ cd ~/dj4e-samples
$ python3 manage.py makemigrations
Migrations for 'users':
users/migrations/0001_initial.py
    - Create model User

$ python3 manage.py migrate
Running migrations:
Applying contenttypes.0001_initial... OK
...
Applying sessions.0001_initial... OK
Applying users.0001_initial... OK
```

models.py:

```
from django.db import models

class User(models.Model):
    name = models.CharField(max_length=128)
    email = models.CharField(max_length=128)
```

https://github.com/csev/dj4e-samples/tree/master/users

Inserting a Record

```
$ cd ~/dj4e-samples
$ python3 manage.py shell
>>> from users.models import User
>>> u = User(name='Kristen', email='kf@umich.edu')
>>> u.save()
>>> print(u.id)
1
>>> print(u.email)
kf@umich.edu
>>>
```

```
INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')
```

https://github.com/csev/dj4e-samples/tree/master/users

CRUD in the ORM

```
u = User(name='Sally', email='a2@umich.edu')
u.save()

User.objects.values()
User.objects.filter(email='csev@umich.edu').values()

User.objects.filter(email='ted@umich.edu').delete()
User.objects.values()

User.objects.filter(email='csev@umich.edu').update(name='Charles')
User.objects.values()

User.objects.values().order_by('email')
User.objects.values().order_by('-name')
```

Model Field Types

- AutoField
- BigAutoField
- BigIntegerField
- BinaryField
- BooleanField
- CharField
- DateField
- DateTimeField
- DecimalField
- DurationField
- EmailField
- FileField
- FilePathField
- FloatField
- GeneratedField
- GenericIPAddressField
- ImageField
- IntegerField
- JSONField
- PositiveBigIntegerField
- PositiveIntegerField
- PositiveSmallIntegerField
- SlugField
- SmallAutoField
- SmallIntegerField
- TextField
- TimeField
- URLField
- UUIDField
- ForeignKey
- ManyToManyField
- OneToOneField

https://docs.djangoproject.com/en/5.0/ref/models/fields/#field-types

Models, Migrations, and Database Tables

Migrations: From Model to Database

- The makemigrations command reads all the models.py files in all the applications, end creates / evolves the migration files
- Guided by the applications listed in settings.py
- Migrations are portable across databases

- The migrate command reads all the migrations folders in the application folders and creates / evolves the tables in the database (i.e. db.sqlite3)

makemigrations

```
dj4e-samples$ ls */models.py
autos/models.py          many/models.py
bookone/models.py        menu/models.py
crispy/models.py         myarts/models.py
favs/models.py           pics/models.py
favsql/models.py         rest/models.py
form/models.py           route/models.py
forums/models.py         session/models.py
getpost/models.py        tmpl/models.py
gview/models.py          tracks/models.py
hello/models.py          users/models.py
home/models.py           views/models.py
dj4e-samples$
```

```
dj4e-samples$ ls  */migrations/0*.py
autos/migrations/0001_initial.py
bookmany/migrations/0001_initial.py
bookone/migrations/0001_initial.py
favs/migrations/0001_initial.py
favsql/migrations/0001_initial.py
forums/migrations/0001_initial.py
gview/migrations/0001_initial.py
many/migrations/0001_initial.py
myarts/migrations/0001_initial.py
pics/migrations/0001_initial.py
rest/migrations/0001_initial.py
tracks/migrations/0001_initial.py
users/migrations/0001_initial.py
dj4e-samples$
```

migrate

```
dj4e-samples$ ls  */migrations/0*.py
autos/migrations/0001_initial.py
bookmany/migrations/0001_initial.py
bookone/migrations/0001_initial.py
favs/migrations/0001_initial.py
favsql/migrations/0001_initial.py
forums/migrations/0001_initial.py
gview/migrations/0001_initial.py
many/migrations/0001_initial.py
myarts/migrations/0001_initial.py
pics/migrations/0001_initial.py
rest/migrations/0001_initial.py
tracks/migrations/0001_initial.py
users/migrations/0001_initial.py
dj4e-samples$
```

```
dj4e-samples$ sqlite3 db.sqlite3
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
sqlite> .tables
auth_group                  gview_car
auth_group_permissions      gview_cat
auth_permission             gview_dog
auth_user                   gview_horse
auth_user_groups            many_course
auth_user_user_permissions  many_membership
autos_auto                  many_person
autos_make                  myarts_article
bookone_book                pics_pic
bookone_instance            rest_breed
bookone_lang                rest_cat
django_admin_log            social_auth_association
django_content_type         social_auth_code
django_migrations           social_auth_nonce
django_session              social_auth_partial
favs_fav                    social_auth_usersocialauth
favs_thing                  tracks_album
favsql_fav                  tracks_artist
favsql_thing                tracks_genre
forums_comment              tracks_track
forums_forum                users_user
sqlite> .quit
dj4e-samples$
```

Prática 02

- Vamos criar nossas primeira classes de modelo

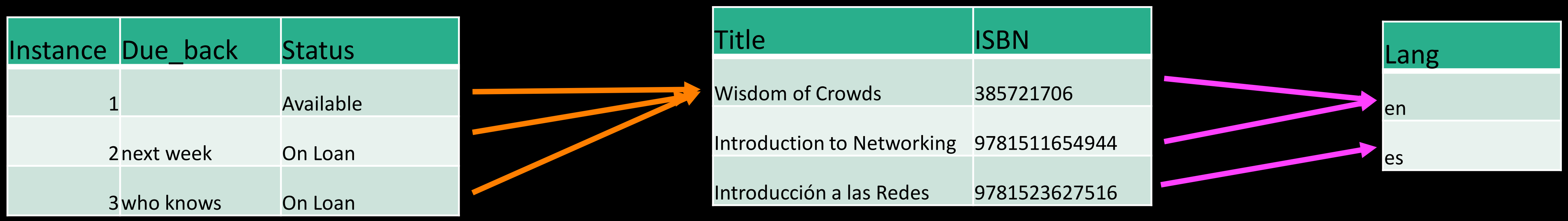
Representing Links (Relationships) in Django

Lets get our ORM on…

Model Field Types

- AutoField
- BigAutoField
- BigIntegerField
- BinaryField
- BooleanField
- CharField
- DateField
- DateTimeField
- DecimalField
- DurationField
- EmailField
- FileField
- FilePathField
- FloatField
- GeneratedField
- GenericIPAddressField
- ImageField
- IntegerField
- JSONField
- PositiveBigIntegerField
- PositiveIntegerField
- PositiveSmallIntegerField
- SlugField
- SmallAutoField
- SmallIntegerField
- TextField
- TimeField
- URLField
- UUIDField
- ForeignKey
- ManyToManyField
- OneToOneField

https://docs.djangoproject.com/en/5.0/ref/models/fields/#field-types



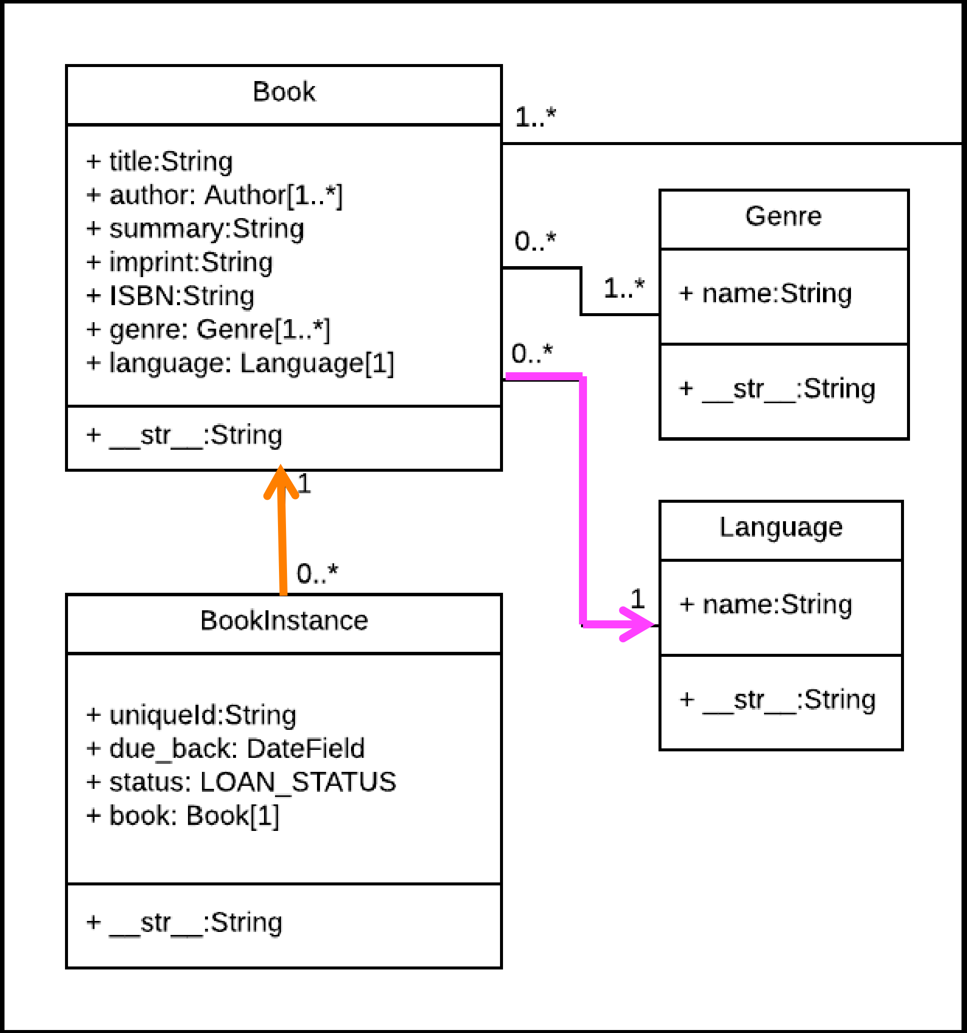
Instance	Due_back	Status
1		Available
2	next week	On Loan
3	who knows	On Loan

Title	ISBN
Wisdom of Crowds	385721706
Introduction to Networking	9781511654944
Introducción a las Redes	9781523627516

Lang
en
es

```
from django.db import models

class Lang(models.Model):
    name = models.CharField(max_length=200)

class Book(models.Model):
    title = models.CharField(max_length=200)
    isbn = models.CharField(max_length=13)
    lang = models.ForeignKey('Lang', on_delete=models.SET_NULL, null=True)

class Instance(models.Model):
    book = models.ForeignKey('Book', on_delete=models.CASCADE)
    due_back = models.DateField(null=True, blank=True)
```

```python
from django.db import models

class Lang(models.Model):
    name = models.CharField(max_length=200)

class Book(models.Model):
    title = models.CharField(max_length=200)
    isbn = models.CharField(max_length=13)
    lang = models.ForeignKey('Lang',
        on_delete=models.SET_NULL, null=True)

class Instance(models.Model):
    due_back = models.DateField(null=True, blank=True)
    book = models.ForeignKey('Book',
        on_delete=models.CASCADE)
```



https://github.com/csev/dj4e-samples/blob/master/bookone/models.py

From Model to Database

```
$ python3 manage.py makemigrations
Migrations for 'bookone':
  bookone/migrations/0001_initial.py
    - Create model Book
    - Create model Instance
    - Create model Lang
    - Add field lang to book

$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin,...
Running migrations:
  Applying bookone.0001_initial... OK
```

Note that makemigrations only "does something" when you create or alter a models.py file. The migrate only "does something" when there are migrations that are not yet applied to the database. Also an application must be added to settings.py before these commands see the models.py file for an application.

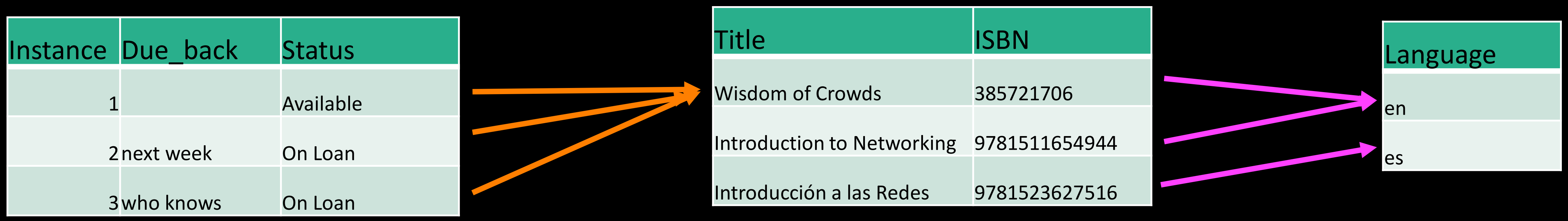
About on_delete

- What do we do when a row in one table points to a row in a "foreign" table via a foreign key and the "destination row" is deleted
  - on_delete = set_null – Keep the row but set foreign key to null
  - on_delete = cascade - Delete the row

id	Title	ISBN	lang_id
1	Wisdom of Crowds	385721706	1
2	Introduction to Networking	9781511654944	1
3	Introducción a las Redes	9781523627516	2

id	Lang
~~1~~	~~en~~
2	es

https://docs.djangoproject.com/en/4.2/ref/models/fields/#django.db.models.ForeignKey.on_delete



Instance	Due_back	Status
1		Available
2	next week	On Loan
3	who knows	On Loan

Title	ISBN
Wisdom of Crowds	385721706
Introduction to Networking	9781511654944
Introducción a las Redes	9781523627516

Language
en
es

```
from django.db import models

class Lang(models.Model):
    name = models.CharField(max_length=200)

class Book(models.Model):
    title = models.CharField(max_length=200)
    isbn = models.CharField(max_length=13)
    lang = models.ForeignKey('Lang', on_delete=models.SET_NULL, null=True)

class Instance(models.Model):
    book = models.ForeignKey('Book', on_delete=models.CASCADE)
    due_back = models.DateField(null=True, blank=True)
```

Prática 03

- Vamos criar nossos primeiros relacionamentos no modelo

Making the super user

- We need to "bootstrap" our system and make a user that can log into the admin page and make more users

```
dj4e-samples$ python3 manage.py createsuperuser
Username: stackadmin
Email address: <seulogin>@cesar.school
Password:
Password (again):
Superuser created successfully.
```

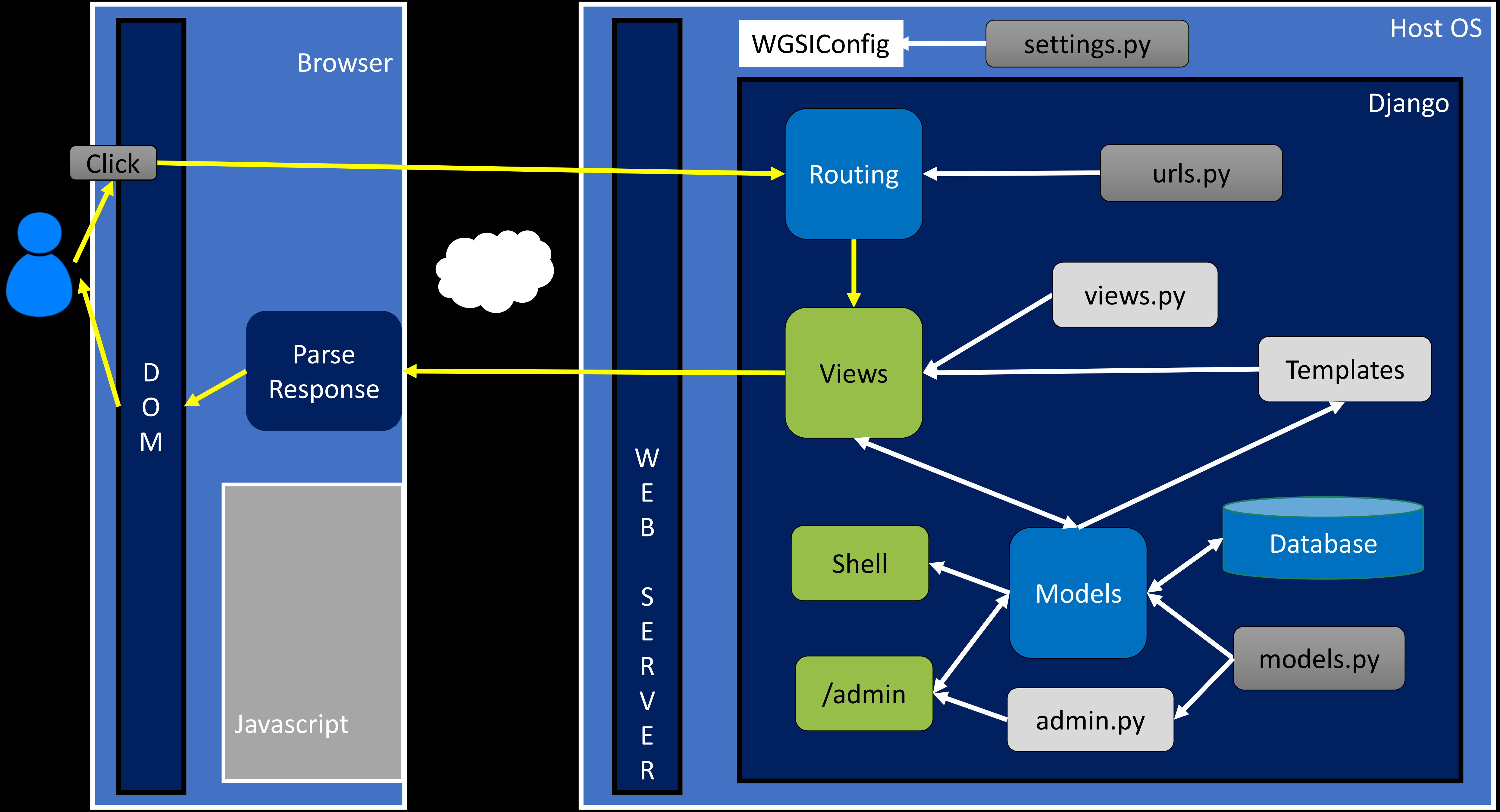
Prática 04

- Vamos criar o *super usuário* e manipular os objetos via interface de administração do Django

Charles Severance

www.dj4e.com

Views and Templates


Browser
Click
D O M
Parse Response
Javascript
W E B S E R V E R
Host OS
WGSIConfig
settings.py
Django
Routing
urls.py
views.py
Views
Templates
Shell
Models
Database
/admin
models.py
admin.py

Views are the core of our application

- Django looks at the incoming request URL and uses urls.py to select a view
- The view from views.py
  - Handle any incoming data in the request and copy it to the database through the model
  - Retrieve data to put on the page from the database though the model
  - Produce the HTML that will become the response and return it to the browser

`https://samples.dj4e.com/`

Reading the URL

- When Django receives an HTTP request it parses it, uses some of the URL for routing purposes and passes parts of the URL to your code

Django Application (also folder)

View within application

```
https://samples.dj4e.com/views/funky
```

Key / value parameter (GET)

```
https://samples.dj4e.com/views/danger?guess=42
```

```
https://samples.dj4e.com/views/rest/24
```

URL Path Parameter

URL Dispatcher

A clean, elegant URL scheme is an important detail in a high-quality web application. Django lets you design URLs however you want, with no framework limitations.

To design URLs for an app, you create a Python module informally called a URLconf (URL configuration). This module is pure Python code and is a mapping between URL path expressions to Python functions (your views).

This mapping can be as short or as long as needed. It can reference other mappings. And, because it’s pure Python code, it can be constructed dynamically.

https://docs.djangoproject.com/en/5.0/topics/http/urls/

Three patterns for views (in urls.py)

- Requests are routed to a pre-defined class from Django itself
- Requests are routed to a function in views.py that takes the http request as a parameter and returns a response
- Requests are routed to a class in views.py that has get() and post() methods that take the http request as a parameter and return a response

views/urls.py

```python
from django.urls import path
from . import views
from django.views.generic import TemplateView

# https://docs.djangoproject.com/en/4.2/topics/http/urls/
app_name='views'
urlpatterns = [
    # pre-defined class from Django
    path('', TemplateView.as_view(template_name='views/main.html')),
    # function from views.py
    path('funky', views.funky),
    path('danger', views.danger),
    path('game', views.game),
    path('rest/<int:guess>', views.rest),
    path('bounce', views.bounce),
    # our class from views.py
    path('main', views.MainView.as_view()),
    path('remain/<slug:guess>', views.RestMainView.as_view()),
]
```

Viewing the Views

```
path('', TemplateView.as_view(template_name='views/main.html'))
```

views/urls.py

views/templates/views/main.html

```
<html><body><p>This is the views main.html sample</p>
<p>
<ul>
    <li>This page is coming from a file in views/templates/main.html</li>
    <li><a href="funky">Use a view function</a></li>
        ...
</ul>
</p>
<p>This sample code is available at
<a href="https://github.com/csev/dj4e-samples" target="_blank">
https://github.com/csev/dj4e-samples</a>
</p>
</body></html>
```

Request and Response Objects

Django uses request and response objects to pass information throughout your Django application.

When a page is requested by a browser, Django creates an **HttpRequest** object that contains metadata about the request.

Then Django loads the appropriate view, passing the **HttpRequest** as the first argument to the view function. Each view is responsible for returning an **HttpResponse** object.

The Application Programming Interfaces (APIs) for **HttpRequest** and **HttpResponse** objects, are defined in the **django.http** module.

https://docs.djangoproject.com/en/5.0/ref/request-response/

class HttpRequest

Attributes
All attributes should be considered read-only, unless stated otherwise.

HttpRequest.scheme
A string representing the scheme of the request (http or https usually).

HttpRequest.body

The raw HTTP request body as a bytestring. This is useful for processing data in different ways than conventional HTML forms: binary images, XML payload etc. For processing conventional form data, use HttpRequest.POST.

https://docs.djangoproject.com/en/5.0/ref/request-response/#httprequest-objects

class HttpResponse

In contrast to HttpRequest objects, which are created automatically by Django, HttpResponse objects are your responsibility.

Each view you write is responsible for instantiating, populating, and returning an HttpResponse.

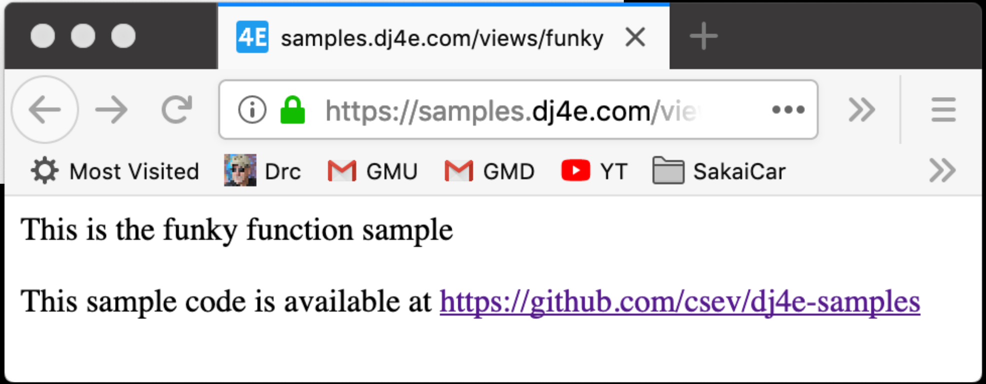
Passing strings

Typical usage is to pass the contents of the page, as a string or bytestring, to the HttpResponse constructor.

https://docs.djangoproject.com/en/5.0/ref/request-response/#httpresponse-objects https://samples.dj4e.com/views/funky

```
path('funky', views.funky),
```

```
from django.http import HttpResponse
from django.http import HttpResponseRedirect

# Create your views here.

def funky(request):
    response = """<html><body><p>This is the funky function sample</p>
    <p>This sample code is available at
    <a href="https://github.com/csev/dj4e-samples">
    https://github.com/csev/dj4e-samples</a></p>
    </body></html>"""
    return HttpResponse(response)
```



This is the funky function sample

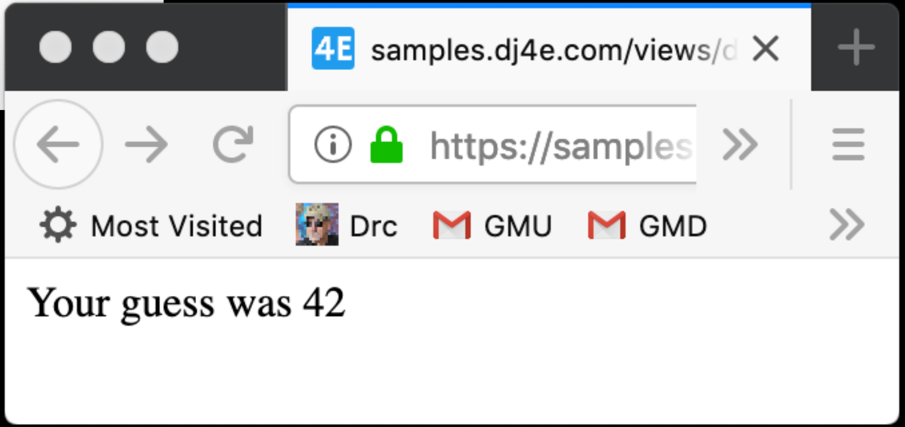
This sample code is available at https://github.com/csev/dj4e-samples https://samples.dj4e.com/views/guess?guess=42

```
path('guess', views.guess),
```

views/urls.py

```
from django.http import HttpResponse
from django.http import HttpResponseRedirect

# Create your views here.

def guess(request) :
    response = """<html><body>
    <p>Your guess was """+request.GET['guess']+"""</p>
    </body></html>"""
    return HttpResponse(response)
```

views/views.py



Your guess was 42

Parsing the URL after the Application and View

https://samples.dj4e.com/views/rest/41

```
urlpatterns = [
    path('rest/<int:guess>', views.rest),
]
```

<type:parameter-name>

```
from django.http import HttpResponse
from django.utils.html import escape

def rest(request, guess) :
    response = """<html><body>
    <p>Your guess was """+escape(guess)+"""</p>
    </body></html>"""
    return HttpResponse(response)
```

Class Views – Inheritance

```
path('main', views.MainView.as_view()),
```

```
from django.http import HttpResponse
from django.utils.html import escape
from django.views import View

class MainView(View) :
    def get(self, request):
        response = """<html><body><p>Hello world MainView in HTML</p>
        <p>This sample code is available at
        <a href="https://github.com/csev/dj4e-samples">
        https://github.com/csev/dj4e-samples</a></p>
        </body></html>"""
        return HttpResponse(response)
```

https://www.py4e.com/lessons/Objects

Parameters to Class Views

`https://samples.dj4e.com/views/remain/abc123-42-xyzzy`

```
path('remain/<slug:guess>', views.RestMainView.as_view()),
```

```
from django.http import HttpResponse
from django.utils.html import escape
from django.views import View

class RestMainView(View) :
    def get(self, request, guess):
        response = """<html><body>
        <p>Your guess was """+escape(guess)+"""</p>
        </body></html>"""
        return HttpResponse(response)
```

HTTP Status Codes

- http://www.dr-chuck.com/page1.htm - 200 OK
- http://www.dj4e.com/nowhere.htm - 404 Not Found
- 500 Server Error
- http://www.drchuck.com/ - 302 Found / Moved

    Also known as “redirect”

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

HTTP Location Header

- You can send a "Redirect" response instead of a page response to communicate a "Location:" header to the browser
- The location header includes a URL that the browser is supposed to forward itself to.
- It was originally used for web sites that moved from one URL to another.

http://en.wikipedia.org/wiki/URL_redirection

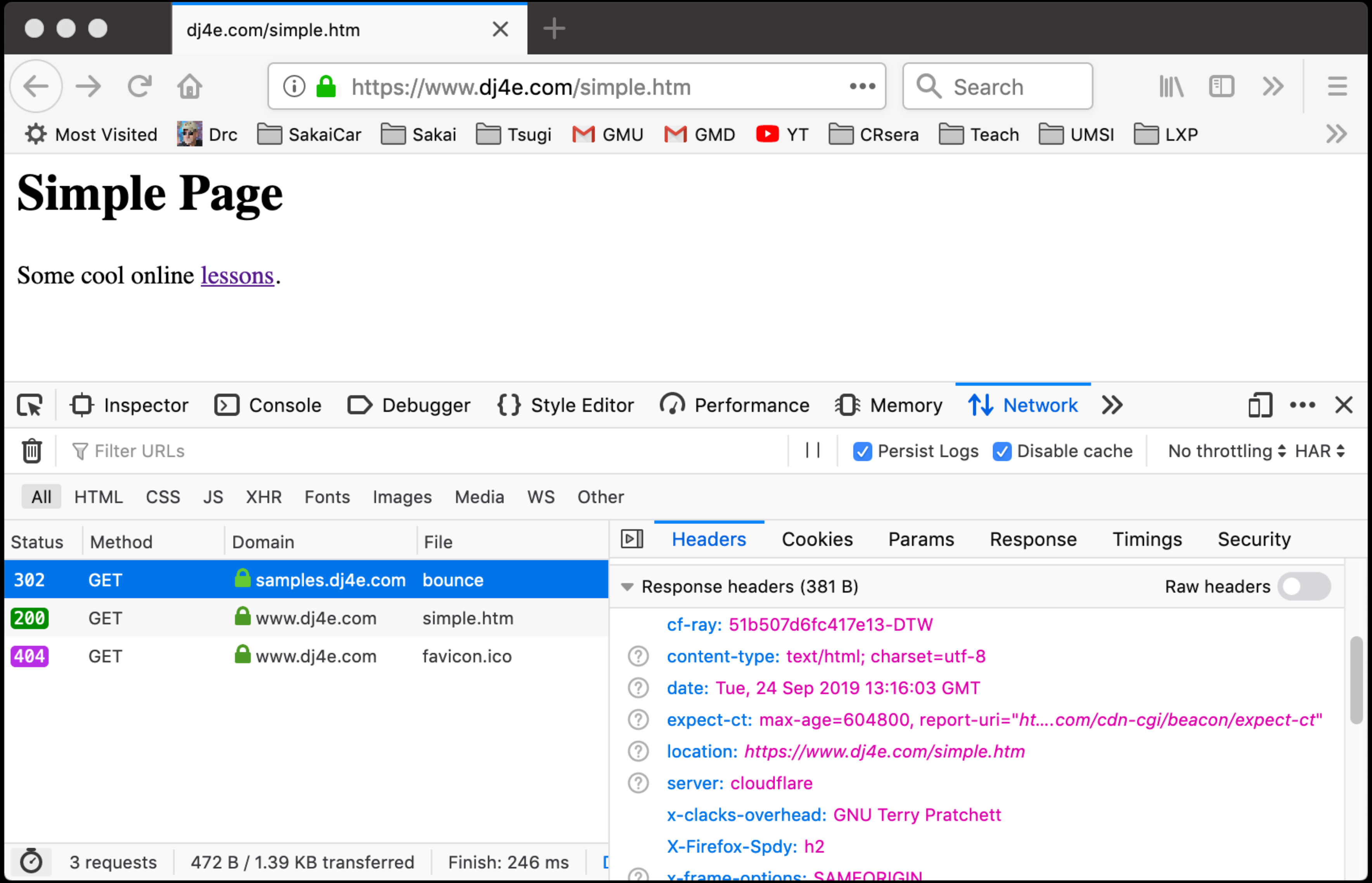
Sending a Redirect from a View

https://samples.dj4e.com/views/bounce

```
 path('bounce', views.bounce)
```

```
from django.http import HttpResponse
from django.http import HttpResponseRedirect

# This is a command to the browser
def bounce(request) :
    return HttpResponseRedirect('https://www.dj4e.com/simple.htm')
```

https://docs.djangoproject.com/en/5.0/ref/request-response/#django.http.HttpResponseRedirect


dj4e.com/simple.htm

https://www.dj4e.com/simple.htm

Simple Page

Some cool online lessons.

Persist Logs ✓ Disable cache ✓ No throttling HAR

Status	Method	Domain	File
302	GET	samples.dj4e.com	bounce
200	GET	www.dj4e.com	simple.htm
404	GET	www.dj4e.com	favicon.ico

3 requests 472 B / 1.39 KB transferred Finish: 246 ms

Response headers (381 B)

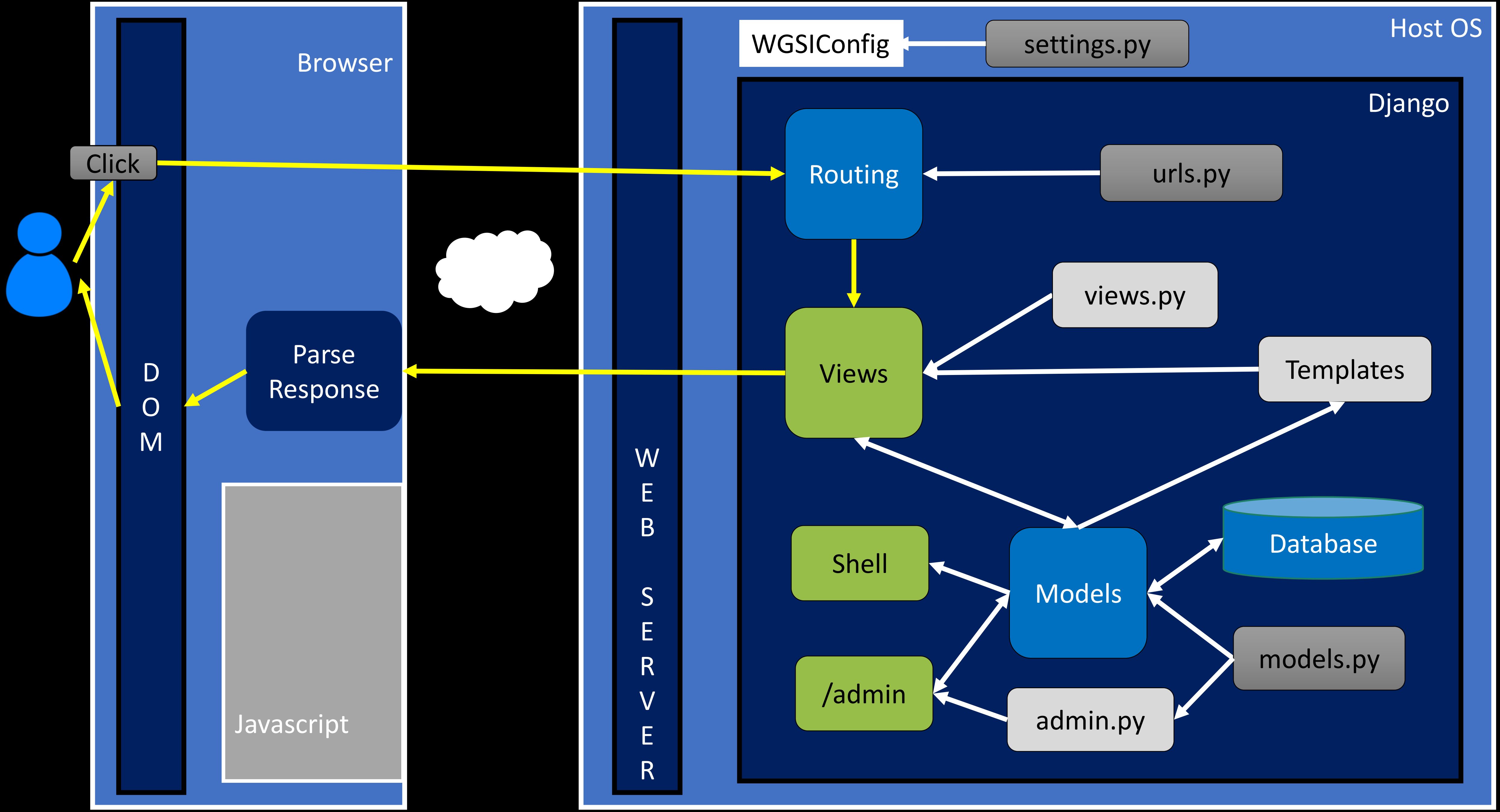
```
cf-ray: 51b507d6fc417e13-DTW
content-type: text/html; charset=utf-8
date: Tue, 24 Sep 2019 13:16:03 GMT
expect-ct: max-age=604800, report-uri="ht....com/cdn-cgi/beacon/expect-ct"
location: https://www.dj4e.com/simple.htm
server: cloudflare
x-clacks-overhead: GNU Terry Pratchett
X-Firefox-Spdy: h2
x-frame-options: SAMEORIGIN
```

Prática 05

- Criando as primeiras versões das *views*
- Nossa aplicação deverá conter as seguintes páginas:
    - Página inicial listando todas as perguntas por ordem inversa de criação
    - Página de detalhes de uma pergunta, listando todas as respostas
    - Opção para votar numa resposta

Templates to Organize HTML

https://github.com/csev/dj4e-samples/tree/master/tmpl


Browser
Click
D O M
Parse Response
Javascript
Host OS
WGSIConfig
settings.py
Django
Routing
urls.py
W E B S E R V E R
views.py
Views
Templates
Shell
Models
Database
/admin
models.py
admin.py

Templates

Being a web framework, Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

A Django project can be configured with one or several template engines (or even zero if you don’t use templates). Django ships built-in backends for its own template system, creatively called the Django template language (DTL), and for the popular alternative Jinja2.

A Django template is a text document marked-up using DTL. Some constructs are recognized and interpreted by the template engine. The main ones are **variables** and **tags**. A template is rendered with a context. Rendering replaces **variables** with their **values**, which are looked up in the **context**, and executes **tags**. Everything else is output as is..

https://docs.djangoproject.com/en/5.0/topics/templates/

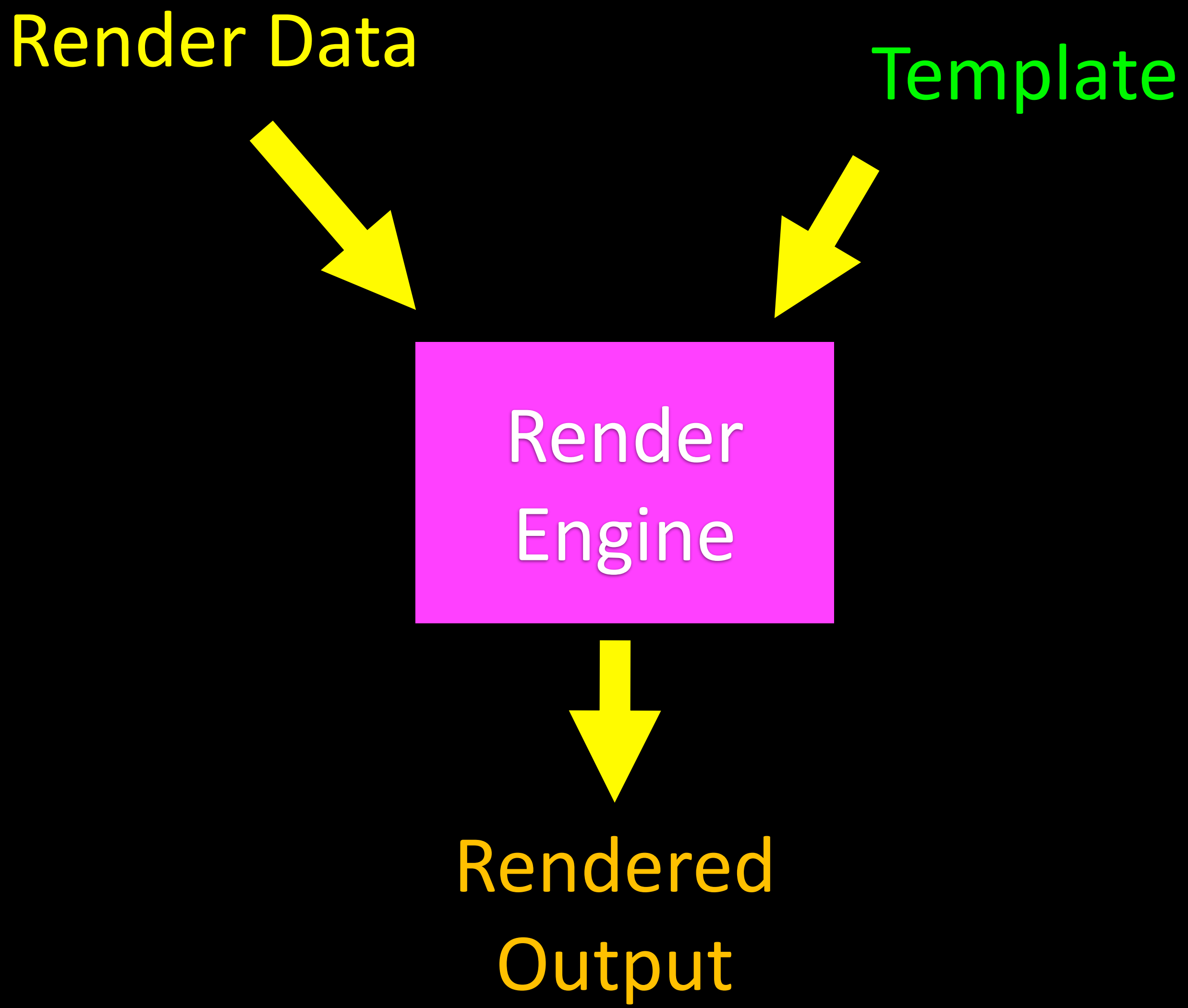
What is a Template?

- Concatenation and escaping can get tiresome and lead to very obtuse looking view code.

```python
from django.http import HttpResponse
from django.utils.html import escape
from django.views import View

class RestMainView(View) :
    def get(self, request, guess):
        response = """<html><body>
        <p>Your guess was """+escape(guess)+"""</p>
        </body></html>"""
        return HttpResponse(response)
```

`https://samples.dj4e.com/views/rest/24`

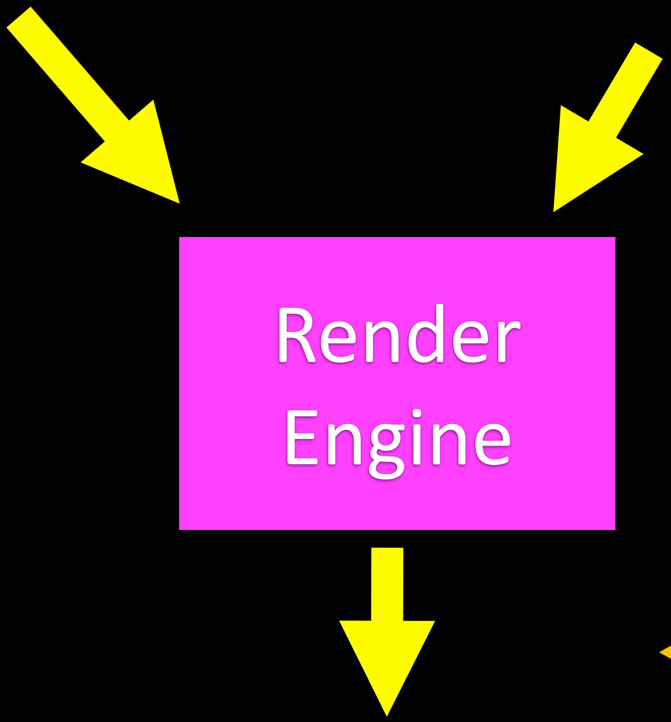
Template Render Process



Template Render Process

```
{ 'dat' : 'Fun > Stuff' }
```

```
<h1>Hi!</h1>
<pre>
{{ dat }}
</pre>
```



```
<h1>Hi!</h1>
<pre>
Fun &gt; Stuff
</pre>
```

URL -> View -> Template

https://samples.dj4e.com/tmpl/game/200

```
path('game/<slug:guess>', views.GameView.as_view())
```

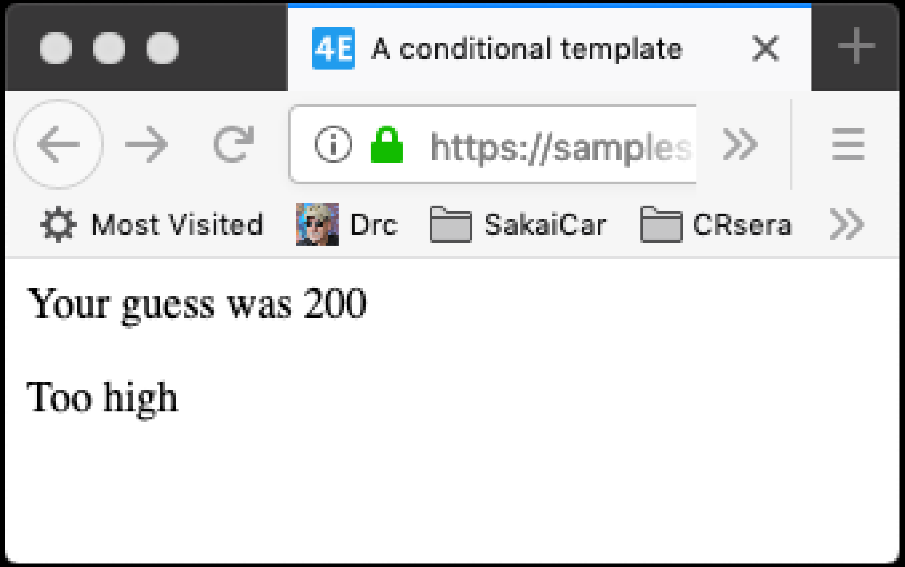
```
from django.shortcuts import render
from django.views import View

class GameView(View) :
    def get(self, request, guess) :
        x = {'guess' : int(guess) }
        return render(request, 'tmpl/cond.html', x)
```

https://samples.dj4e.com/tmpl/game/200

dj4e-samples/tmpl/templates/tmpl/cond.html

```
<html>
<head>
    <title>A conditional template</title>
</head>
<body>
    <p>Your guess was {{ guess }}</p>
    {% if guess < 42 %}
        <p>Too low</p>
    {% elif guess > 42 %}
        <p>Too high</p>
    {% else %}
        <p>Just right</p>
    {% endif %}
</body>
</html>
```

```
from django.views import View

class GameView(View) :
    def get(self, request, guess) :
        x = {'guess' : int(guess) }
        return render(request, 'tmpl/cond.html', x)
```



Your guess was 200

Too high

Where are Templates?

- A Django project is made up of one or more applications in folders

```
dj4e-samples$ ls
LICENSE.md          form                pics
README.md           forums              requirements.txt
autos               getpost             rest
bookmany            gview               route
bookone             hello               scripts
crispy              home                session
db.sqlite3          manage.py           tmpl
dj4e-samples        many                tracks
favs                menu                users
favsql              myarts              views
```

Templates in Folders

- It is common to reuse the "name" of a template file in several applications
- We use a technique called "namespace" so that each application can load its own templates without template name collision

```
dj4e-samples$ ls */templates/*/detail.html favs/templates/favs/detail.html
favsql/templates/favsql/detail.html
forums/templates/forums/detail.html
pics/templates/pics/detail.html
dj4e-samples$
```

https://en.wikipedia.org/wiki/Namespace
https://docs.djangoproject.com/en/4.2/topics/http/urls/#url-namespaces

Templates in Name Spaces

- For the namespace to work, we need to put templates in a path that includes the application name twice. Weird but necessary. ☹

```
dj4e-samples$ ls */templates/*/detail.html favs/templates/favs/detail.html
favsql/templates/favsql/detail.html
forums/templates/forums/detail.html
pics/templates/pics/detail.html
dj4e-samples$
```

https://en.wikipedia.org/wiki/Namespace
https://docs.djangoproject.com/en/5.0/topics/http/urls/#url-namespaces

Django template language (DTL)

https://docs.djangoproject.com/en/5.0/ref/templates/language/

Template Tags / Code

Substitution

```
{{ zap }}
{{ zap|safe }}
```

Calling code

```
{% url 'cat-detail' cat.id %}
{% author.get_absolute_url %}
```

Logic

```
{% if zap > 100 %}
{% endif %}
```

Blocks

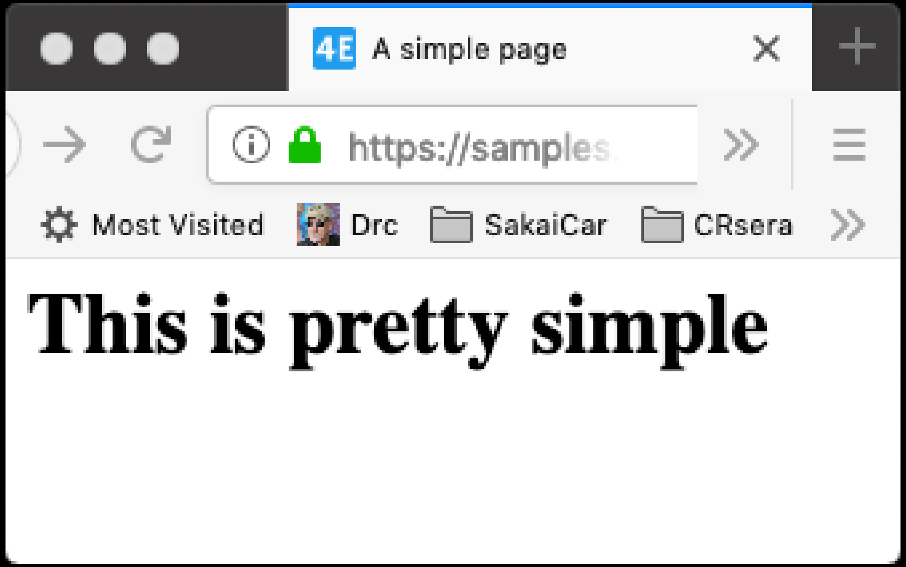
```
{% block content %}
{% endblock %}
```

https://samples.dj4e.com/tmpl/simple

```python
from django.shortcuts import render

def simple(request):
    return render(request, 'tmpl/simple.html')
```

dj4e-samples/tmpl/templates/tmpl/simple.html

```html
<html>
<head>
    <title>A simple page</title>
</head>
<body>
    <h1>This is pretty simple</h1>
</body>
</html>
```



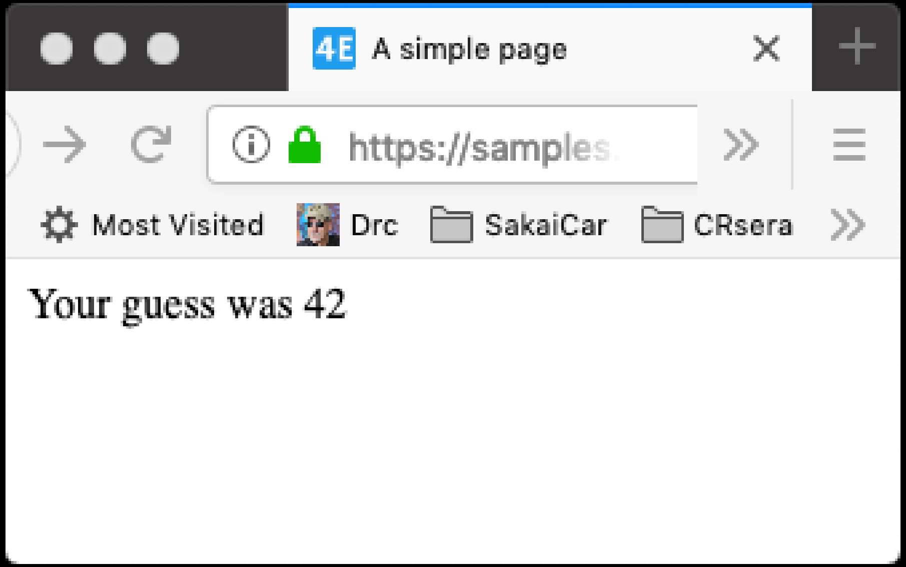
This is pretty simple

https://samples.dj4e.com/tmpl/guess

```
def guess(request) :
    context = {'zap' : '42' }
    return render(request, 'tmpl/guess.html', context)
```

dj4e-samples/tmpl/templates/tmpl/guess.html

```
<html>
<head>
    <title>A simple page</title>
</head>
<body>
    <p>Your guess was {{ zap }}</p>
</body>
</html>
```



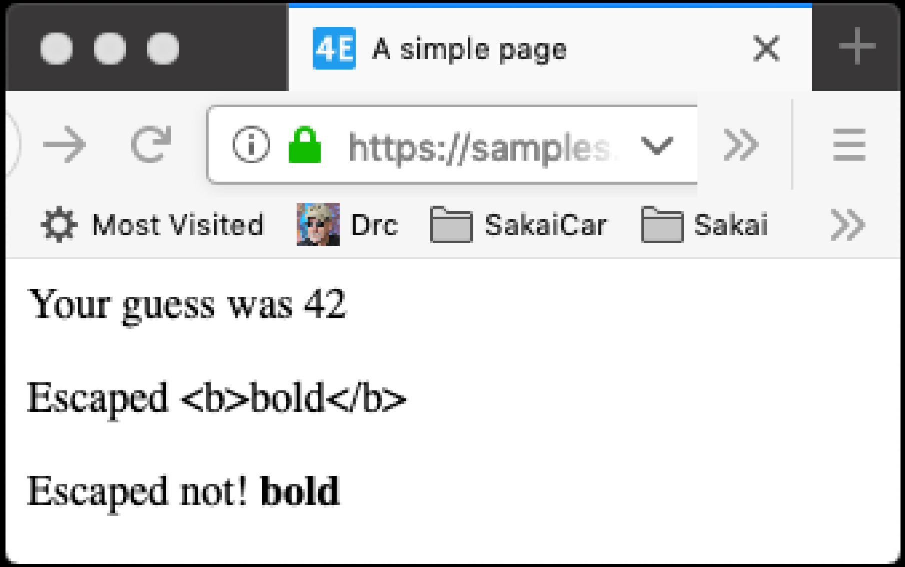
Your guess was 42

https://samples.dj4e.com/tmpl/special

```
def special(request) :
    context = {'txt' : '<b>bold</b>',
               'zap' : '42' }
    return render(request, 'tmpl/special.html', context)
```

dj4e-samples/tmpl/templates/tmpl/special.html

```
<body>
   <p>Your guess was {{ zap }}</p>
    <p>Escaped {{ txt }}</p>
    <p>Escaped not! {{ txt|safe }}</p>
</body>
```



Your guess was 42

Escaped <b>bold</b>

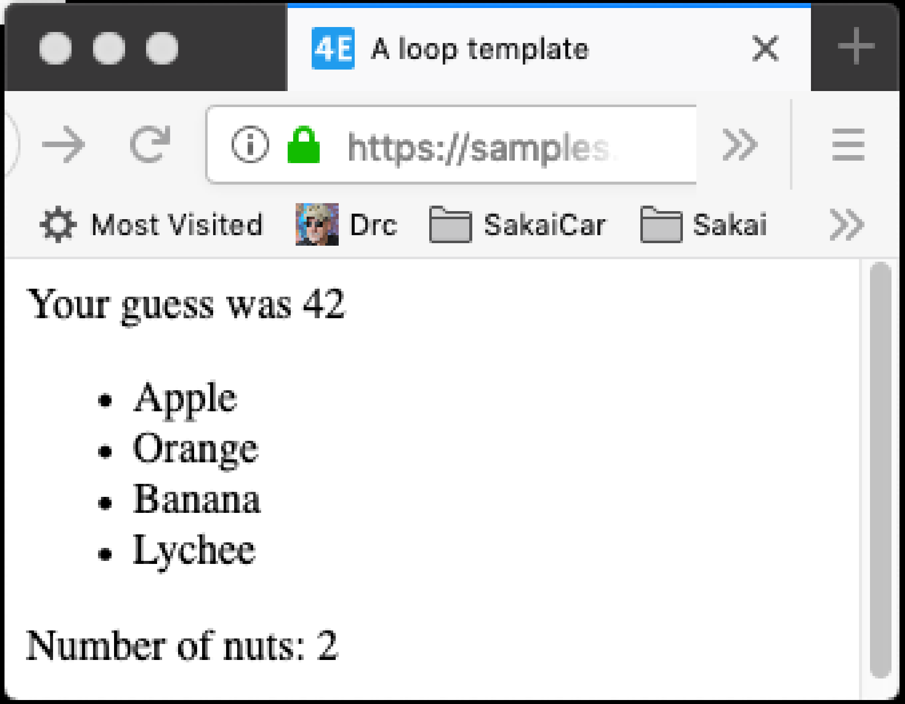
Escaped not! **bold**

https://samples.dj4e.com/tmpl/loop

```
def loop(request) :
    f = ['Apple', 'Orange', 'Banana', 'Lychee']
    n = ['peanut', 'cashew']
    x = {'fruits' : f, 'nuts' : n, 'zap' : '42' }
    return render(request, 'tmpl/loop.html', x)
```

dj4e-samples/tmpl/templates/tmpl/loop.html

```
<ul>
{% for x in fruits  %}
<li>{{ x }}</li>
{% endfor %}
</ul>
{% if nuts %}
    <p>Number of nuts: {{ nuts|length }}</p>
{% else %}
    <p>No nuts.</p>
{% endif %}
```



Your guess was 42

- Apple
- Orange
- Banana
- Lychee

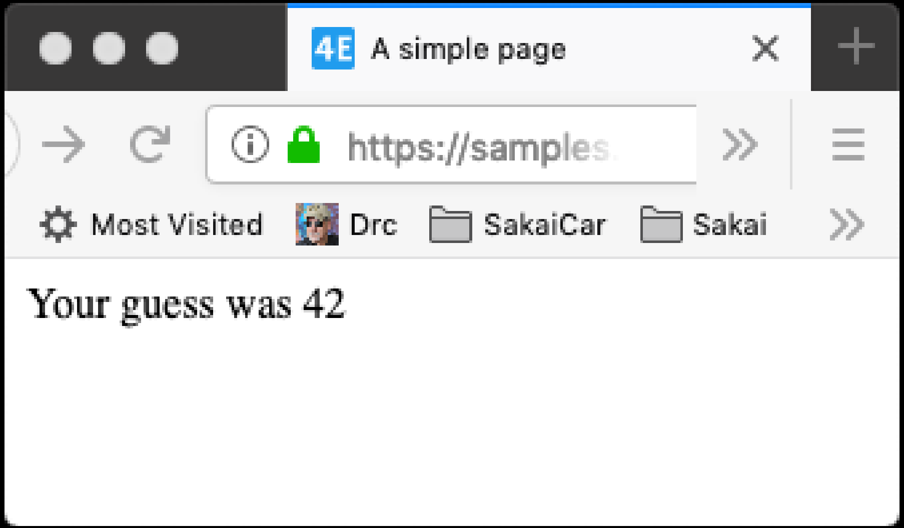
Number of nuts: 2

https://samples.dj4e.com/tmpl/nested

```
def nested(request) :
    x = {'outer' : { 'inner' : '42' } }
    return render(request, 'tmpl/nested.html', x)
```

dj4e-samples/tmpl/templates/tmpl/nested.html

```
<body>
    <p>Your guess was {{ outer.inner }}</p>
</body>
```



Your guess was 42

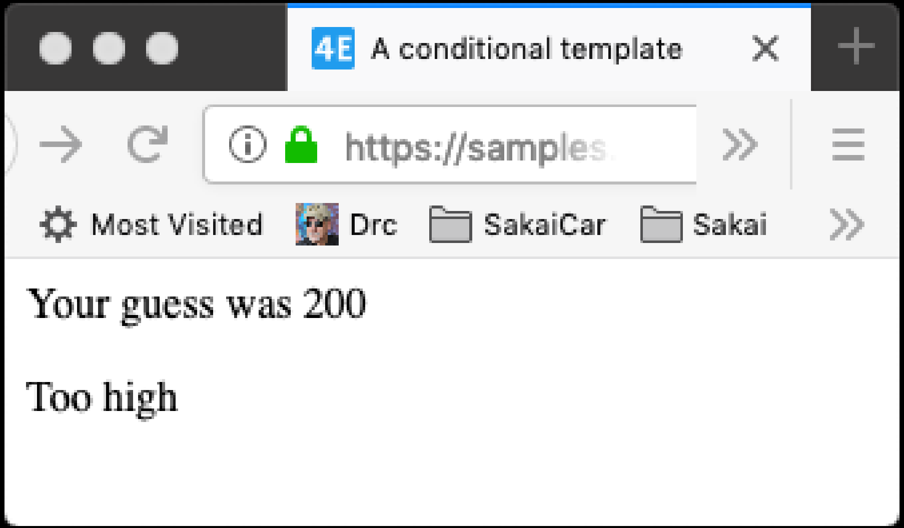
https://samples.dj4e.com/tmpl/game/200

```
path('game/<slug:guess>', views.GameView.as_view())
```

```
class GameView(View) :
    def get(self, request, guess) :
        x = {'guess' : int(guess) }
        return render(request, 'tmpl/cond.html', x)
```

dj4e-samples/tmpl/templates/tmpl/cond.html

```
<p>Your guess was {{ guess }}</p>
{% if guess < 42 %}
    <p>Too low</p>
{% elif guess > 42 %}
    <p>Too high</p>
{% else %}
    <p>Just right</p>
{% endif %}
```



Your guess was 200

Too high

Template Inheritance

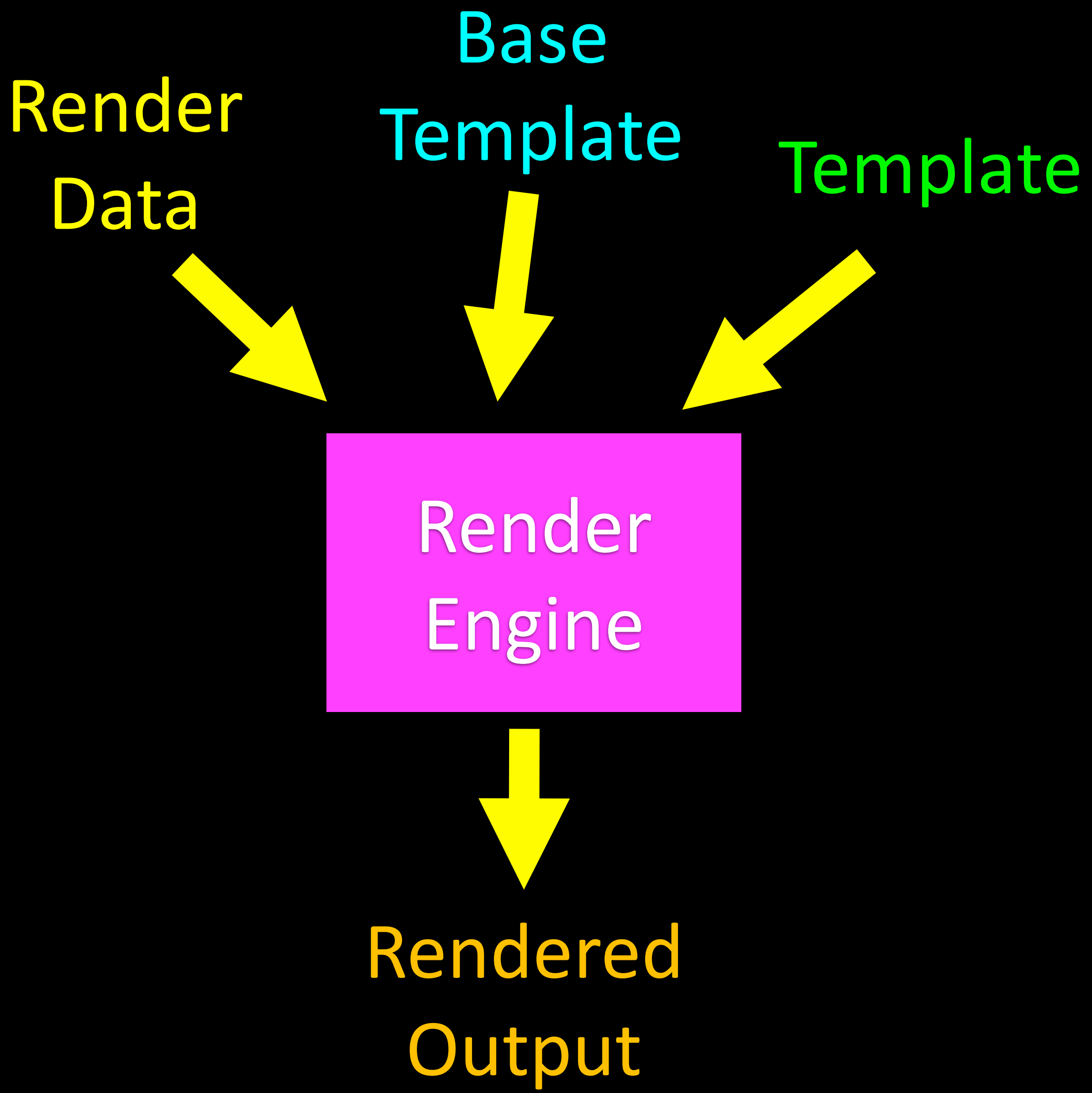
https://docs.djangoproject.com/en/5.0/ref/templates/language/#template-inheritance

Inheritance

- When we make a new template - we can extend an existing template and then add our own little bit to make our new class
- Another form of store and reuse
- Don't Repeat Yourself (DRY)

https://en.wikipedia.org/wiki/Don%27t_repeat_yourself

Template Inheritance



Template Inheritance

tmpl/templates/tmpl/cond.html

```
<html>
<head>
    <title>A conditional template</title>
</head>
<body>
    <p>Your guess was {{ guess }}</p>
    {% if guess < 42 %}
        <p>Too low</p>
    {% elif guess > 42 %}
        <p>Too high</p>
    {% else %}
        <p>Just right</p>
    {% endif %}
</body>
</html>
```

tmpl/templates/tmpl/base.html

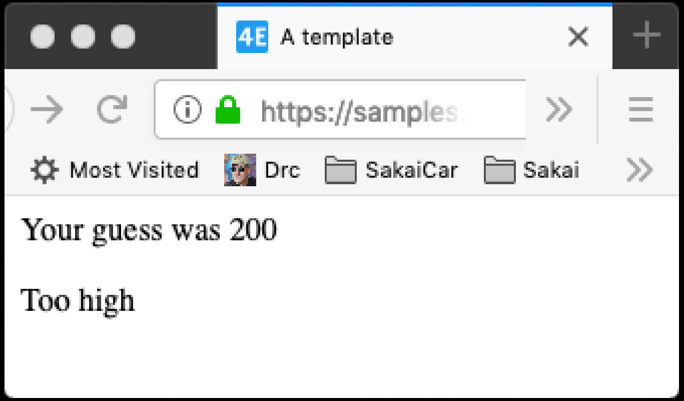
```
<html>
<head>
    <title>Base template</title>
</head>
<body>
    {% block content %}{% endblock %}
</body>
</html>
```

tmpl/templates/tmpl/cond2.html

```
{% extends "tmpl/base.html" %}

{% block content %}
    <p>Your guess was {{ guess }}</p>
    {% if guess < 42 %}
        <p>Too low</p>
    {% elif guess > 42 %}
        <p>Too high</p>
    {% else %}
        <p>Just right</p>
    {% endif %}
{% endblock %}
```

https://samples.dj4e.com/tmpl/game2/200

```
class GameView2(View) :
    def get(self, request, guess) :
        x = {'guess' : int(guess) }
        return render(request, 'tmpl/cond2.html', x)
```



Your guess was 200

Too high

tmpl/templates/tmpl/cond2.html

```
{% extends "tmpl/base.html" %}

{% block content %}
    <p>Your guess was {{ guess }}</p>
    {% if guess < 42 %}
        <p>Too low</p>
    {% elif guess > 42 %}
        <p>Too high</p>
    {% else %}
        <p>Just right</p>
    {% endif %}
{% endblock %}
```

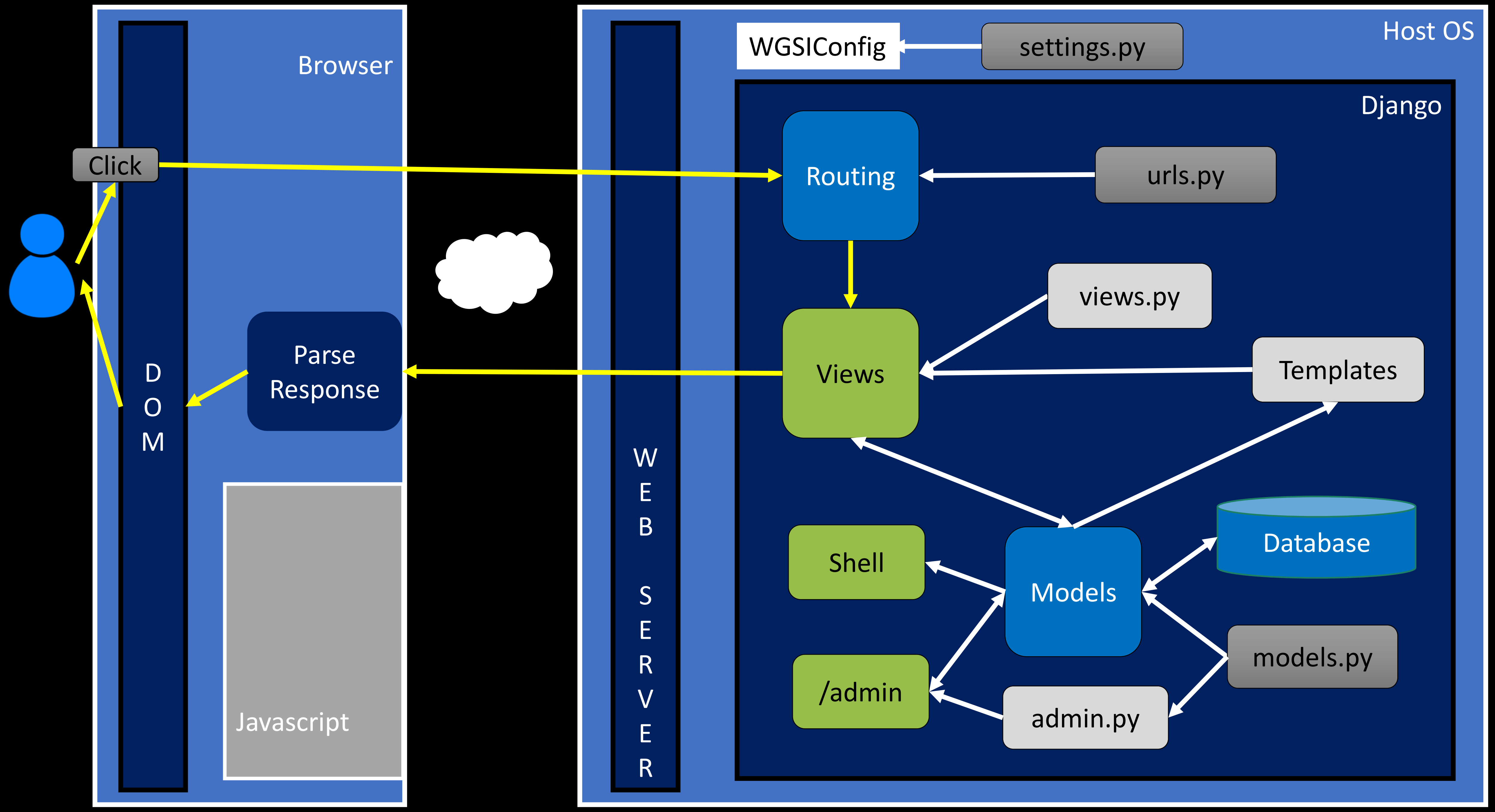
tmpl/templates/tmpl/base.html

```
<html>
<head>
    <title>A template</title>
</head>
<body>
    {% block content %}{% endblock %}
</body>
</html>
```

URL Mapping / Reversing

https://samples.dj4e.com/route/

https://docs.djangoproject.com/en/4.2/topics/http/urls/#reverse-resolution-of-urls


Browser
Click
D O M
Parse Response
Javascript
Host OS
WGSIConfig
settings.py
W E B S E R V E R
Django
Routing
urls.py
views.py
Views
Templates
Shell
Models
Database
/admin
models.py
admin.py

Reverse Resolution of URLs

A common need when working on a Django project is the possibility to obtain URLs in their final forms either for embedding in generated content (views and assets URLs, URLs shown to the user, etc.) or for handling of the navigation flow on the server side (redirections, etc.)

It is strongly desirable to avoid hard-coding these URLs (a laborious, non-scalable and error-prone strategy). Equally dangerous is devising ad-hoc mechanisms to generate URLs that are parallel to the design described by the URLconf, which can result in the production of URLs that become stale over time.

In other words, what's needed is a DRY mechanism. Among other advantages it would allow evolution of the URL design without having to go over all the project source code to search and replace outdated URLs.

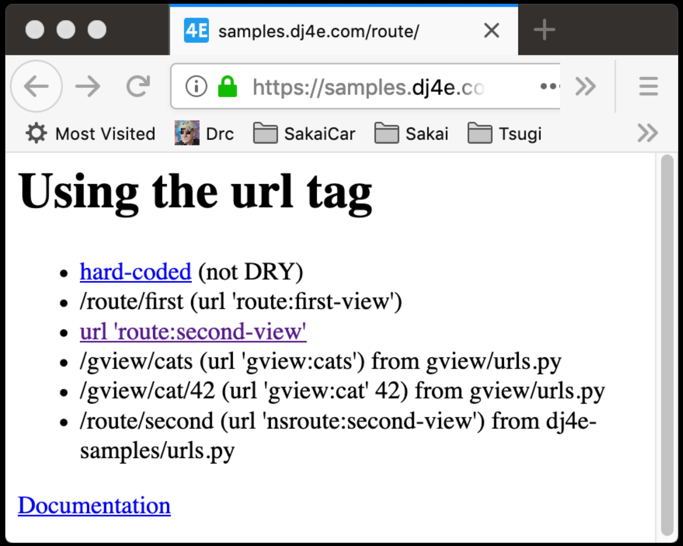
The primary piece of information we have available to get a URL is an identification (e.g. the name) of the view in charge of handling it. Other pieces of information that necessarily must participate in the lookup of the right URL are the types (positional, keyword) and values of the view arguments.

https://docs.djangoproject.com/en/5.0/topics/http/urls/#reverse-resolution-of-urls

dj4e-samples/route/urls.py

```
urlpatterns = [
    path('', TemplateView.as_view(template_name='route/main.html')),
    path('first', views.FirstView.as_view(), name='first-view'),
    path('second', views.SecondView.as_view(), name='second-view'),
]
```

https://samples.dj4e.com/route/

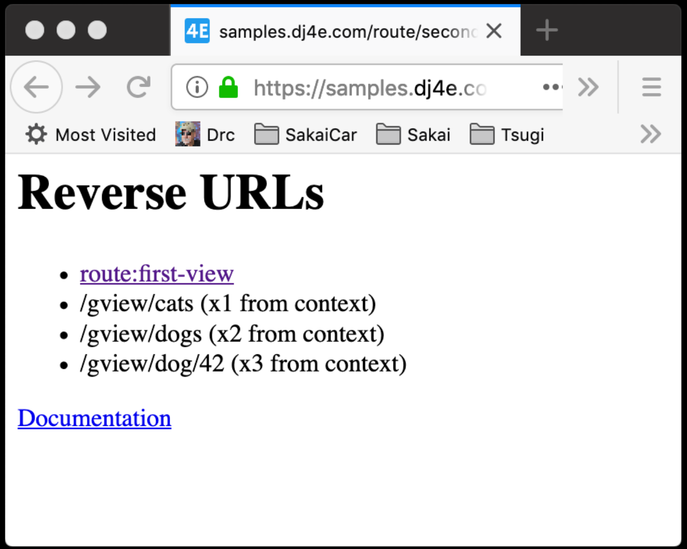


Using the url tag

- hard-coded (not DRY)
- /route/first (url 'route:first-view')
- url 'route:second-view'
- /gview/cats (url 'gview:cats') from gview/urls.py
- /gview/cat/42 (url 'gview:cat' 42) from gview/urls.py
- /route/second (url 'nsroute:second-view') from dj4e-samples/urls.py

Documentation

https://samples.dj4e.com/route/second



Reverse URLs

- route:first-view
- /gview/cats (x1 from context)
- /gview/dogs (x2 from context)
- /gview/dog/42 (x3 from context)

Documentation

```
app_name = 'route'
urlpatterns = [
    path('', TemplateView.as_view(template_name='route/main.html')),
    path('first', views.FirstView.as_view(), name='first-view'),
    path('second', views.SecondView.as_view(), name='second-view'),
]
```

dj4e-samples/route/urls.py

dj4e-samples/route/templates/route/main.html

```
<li>
    <a href="/route/second">
    hard-coded</a> (not DRY)
</li>
<li>
    {% url 'route:first-view' %}
    (url 'route:first-view')
</li>
<li>
    <a href="{% url 'route:second-view' %}">
    url 'route:second-view'</a>
</li>
```

route:first-view

application name

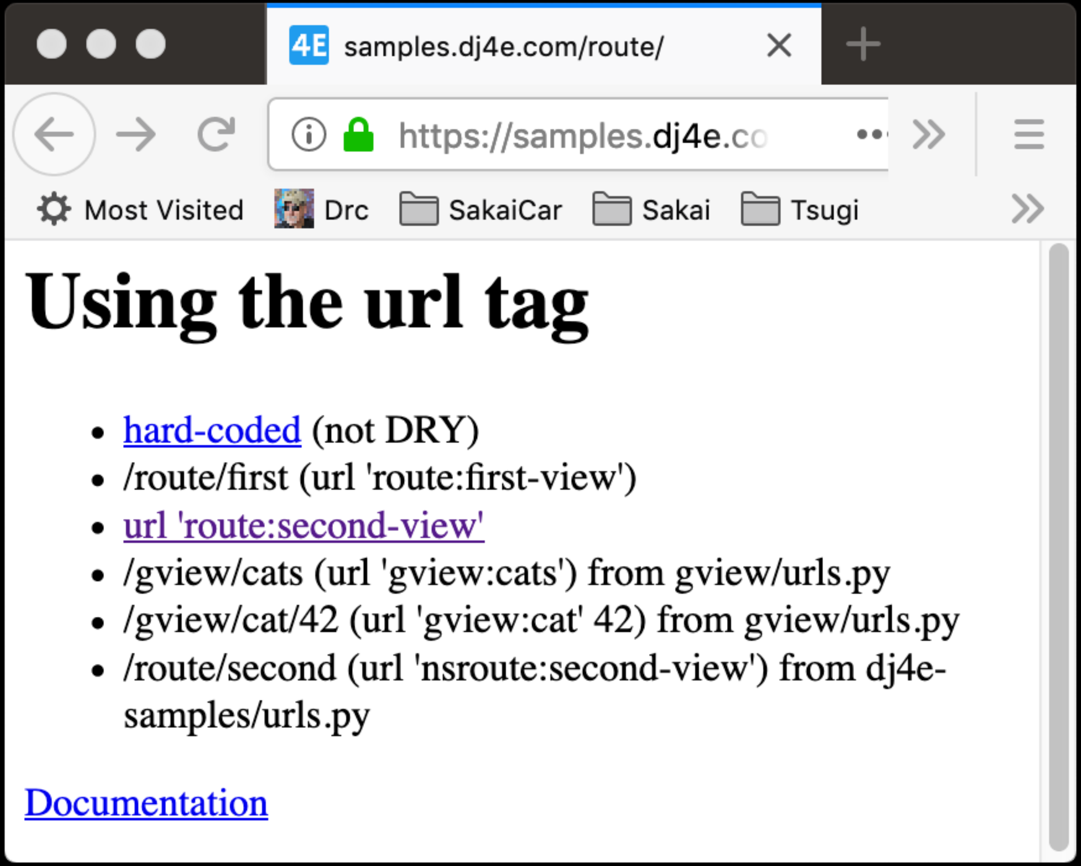
view name

Using the url tag in a template

https://samples.dj4e.com/route/

dj4e-samples/route/templates/route/main.html

```
<li>
    <a href="/route/second-view">
    hard-coded</a> (not DRY)
</li>
<li>
    {% url 'route:first-view' %}
    (url 'route:first-view')
</li>
<li>
    <a href="{% url 'route:second-view' %}">
    url 'route:second-view'</a>
</li>
```



Using the url tag

- hard-coded (not DRY)
- /route/first (url 'route:first-view')
- url 'route:second-view'
- /gview/cats (url 'gview:cats') from gview/urls.py
- /gview/cat/42 (url 'gview:cat' 42) from gview/urls.py
- /route/second (url 'nsroute:second-view') from dj4e-samples/urls.py

Documentation

dj4e-samples/gview/urls.py

```
app_name = 'gview'
urlpatterns = [
    path('cats', views.CatListView.as_view(), name='cats'),
    path('cat/<int:pk_from_url>', views.CatDetailView.as_view(), name='cat'),
]
```

dj4e-samples/route/templates/route/main.html

```
 <li>
    {% url 'gview:cats' %}
   (url 'gview:cats') from gview/urls.py
</li>
<li>
    {% url 'gview:cat' 42 %}
    (url 'gview:cat' 42) from gview/urls.py
</li>
```

Parameter

'gview:cat' 42

application name

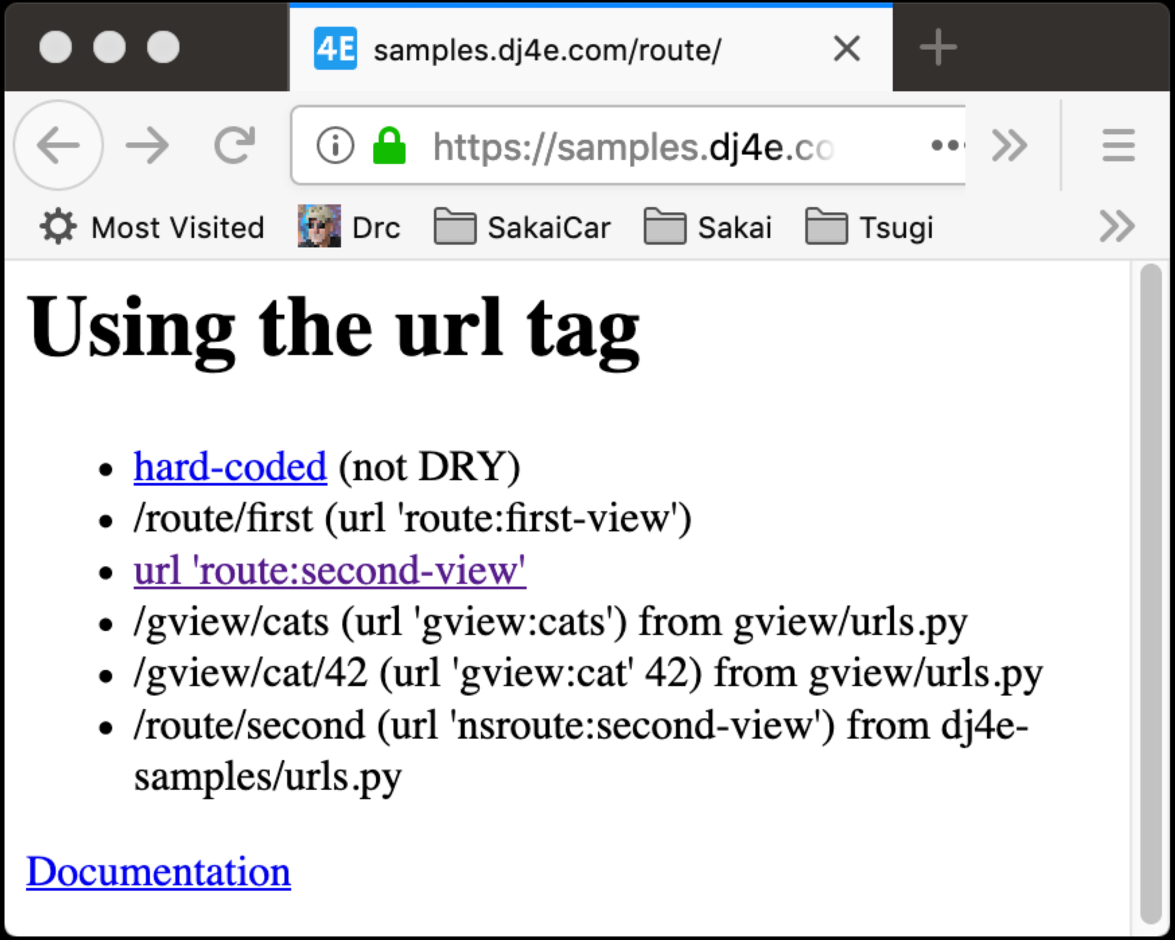
view name

Other applications and parameters

https://samples.dj4e.com/route/

dj4e-samples/route/templates/route/main.html

```
 <li>
    {% url 'gview:cats' %}
   (url 'gview:cats')
</li>
<li>
    {% url 'gview:cat' 42 %}
    (url 'gview:cat' 42)
</li>
```



samples.dj4e.com/route/

https://samples.dj4e.co

Using the url tag

- hard-coded (not DRY)
- /route/first (url 'route:first-view')
- url 'route:second-view'
- /gview/cats (url 'gview:cats') from gview/urls.py
- /gview/cat/42 (url 'gview:cat' 42) from gview/urls.py
- /route/second (url 'nsroute:second-view') from dj4e-samples/urls.py

Documentation

dj4e-samples/dj4e-samples/urls.py

```
urlpatterns = [
    path('', include('home.urls')),
    path('admin/', admin.site.urls),  # Keep
    url(r'^oauth/', include('social_django.urls', namespace='social')),
    path('hello/', include('hello.urls')),
    path('route/', include('route.urls', namespace='nsroute')),
]
```

dj4e-samples/route/templates/route/main.html

```
<li>
  <a href="{% url 'route:second-view' %}">
    url 'route:second-view'</a>
</li>
    ...
<li>
      {% url 'nsroute:second-view' %}
  (url 'nsroute:second-view')
</li>
```

A "second" name space

```
path('second', views.SecondView.as_view(), name='second-view'),
```

dj4e-samples/route/urls.py

dj4e-samples/route/views.py

```
from django.shortcuts import render
from django.urls import reverse
from django.views import View

class SecondView(View):
  def get(self, request) :
    u = reverse('gview:cats')
    u2 = reverse('gview:dogs')
    u3 = reverse('gview:dog', args=['42'] )
    ctx = {'x1' : u, 'x2': u2, 'x3': u3 }
    return render(request, 'route/second.html', ctx)
```

dj4e-samples/route/templates/route/main.html

```
<li>
    <a href="{% url 'route:first-view' %}">
    route:first-view</a>
</li>
<li>
    {{ x1 }} (x1 from context)
</li>
<li>
    {{ x2 }} (x2 from context)
</li>
<li>
    {{ x3 }} (x3 from context)
</li>
```

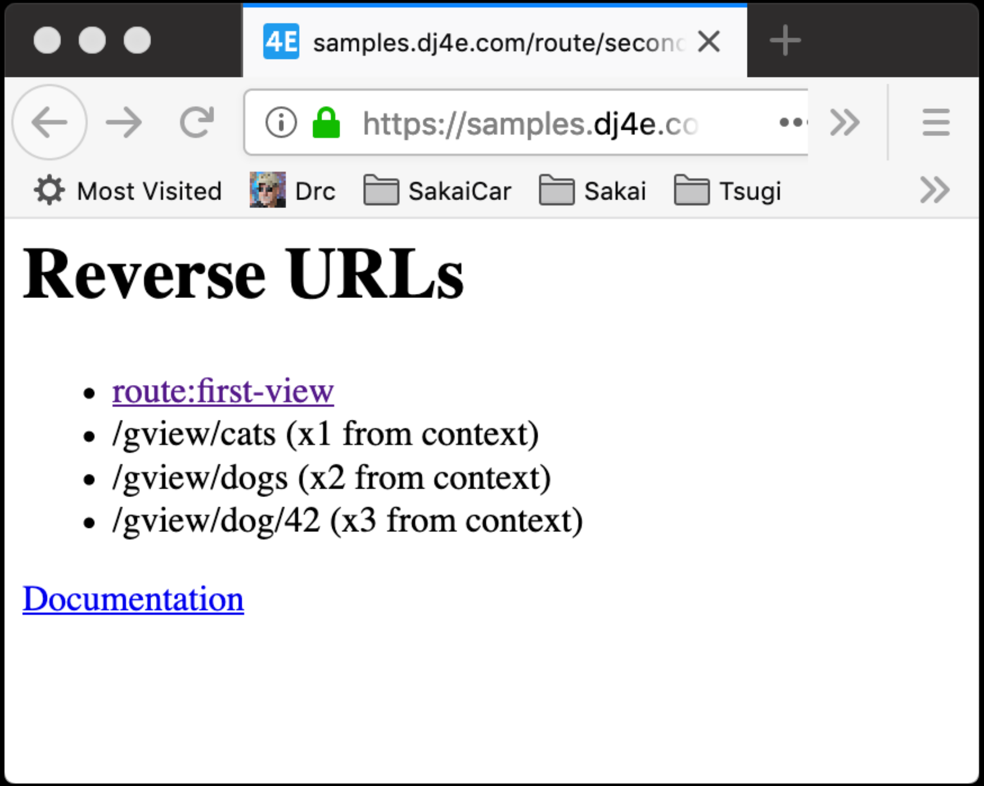
dj4e-samples/route/templates/route/main.html

```
class SecondView(View):
  def get(self, request) :
    u = reverse('gview:cats')
    u2 = reverse ('gview:dogs')
    u3 = reverse('gview:dog', args=['42'] )
    ctx = {'x1' : u, 'x2': u2, 'x3': u3 }
    return render(request, 'route/second.html', ctx)
```

dj4e-samples/route/templates/route/main.html

```
<li>
    {{ x1 }} (x1 from context)
</li>
<li>
    {{ x2 }} (x2 from context)
</li>
<li>
    {{ x3 }} (x3 from context)
</li>
```

https://samples.dj4e.com/route/second



Reverse URLs

- route:first-view
- /gview/cats (x1 from context)
- /gview/dogs (x2 from context)
- /gview/dog/42 (x3 from context)

Documentation

Summary

- Views are where we bring the application components together to handle requests from browsers and produce responses for the browsers
- Templates take a context and merge it into a template to produce HTML
    - Values can be substituted with or without "escaping"
    - Coding in templates

Prática 06

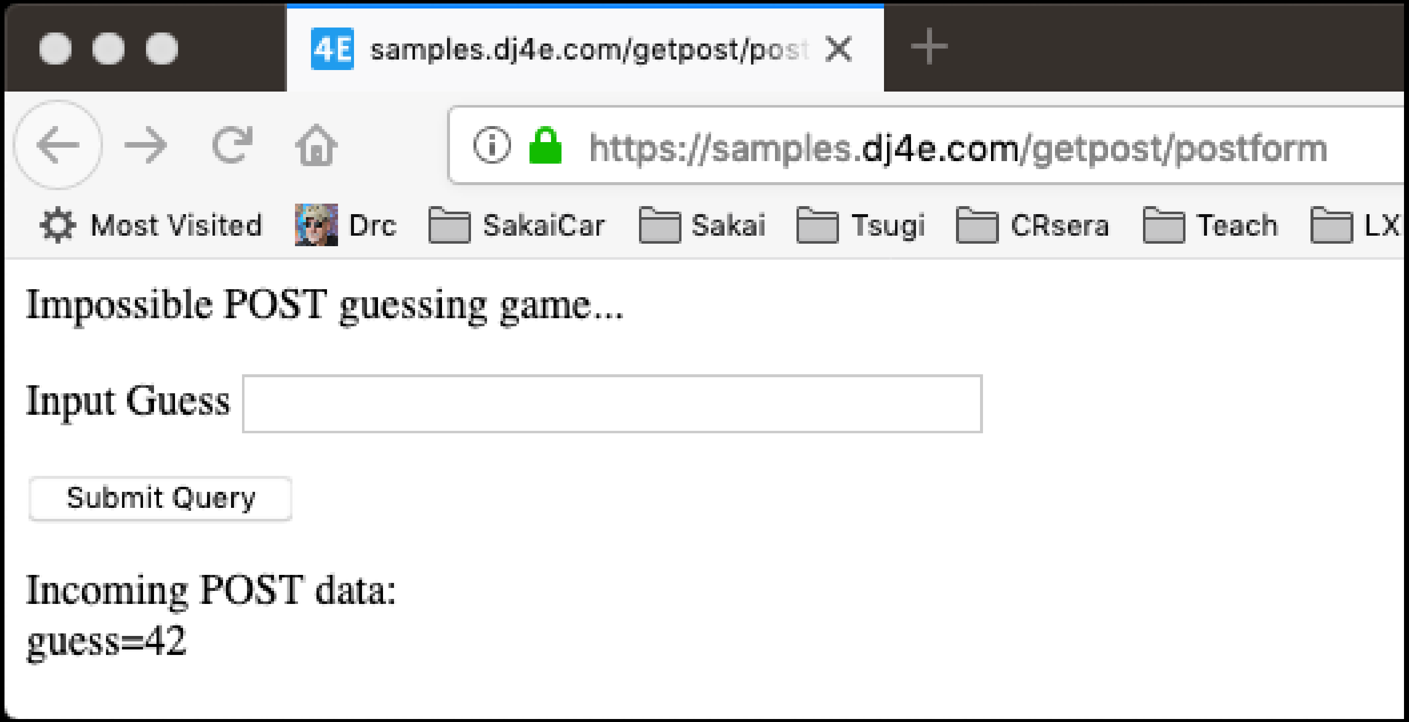
- Criando templates para as views

Charles Severance
www.dj4e.com

Form Processing

https://samples.dj4e.com/getpost/
https://samples.dj4e.com/form/

Forms gather data and send it to the server


samples.dj4e.com/getpost/post
https://samples.dj4e.com/getpost/postform
Most Visited
Drc
SakaiCar
Sakai
Tsugi
CRsera
Teach
LX
Submit Query

Impossible POST guessing game...

Input Guess

Incoming POST data:
guess=42

Forms GET vs. POST

Two ways the browser can send parameters to the web server

- GET - Parameters are placed on the URL which is retrieved.
- POST - The URL is retrieved and parameters are appended to the request in the the HTTP connection.

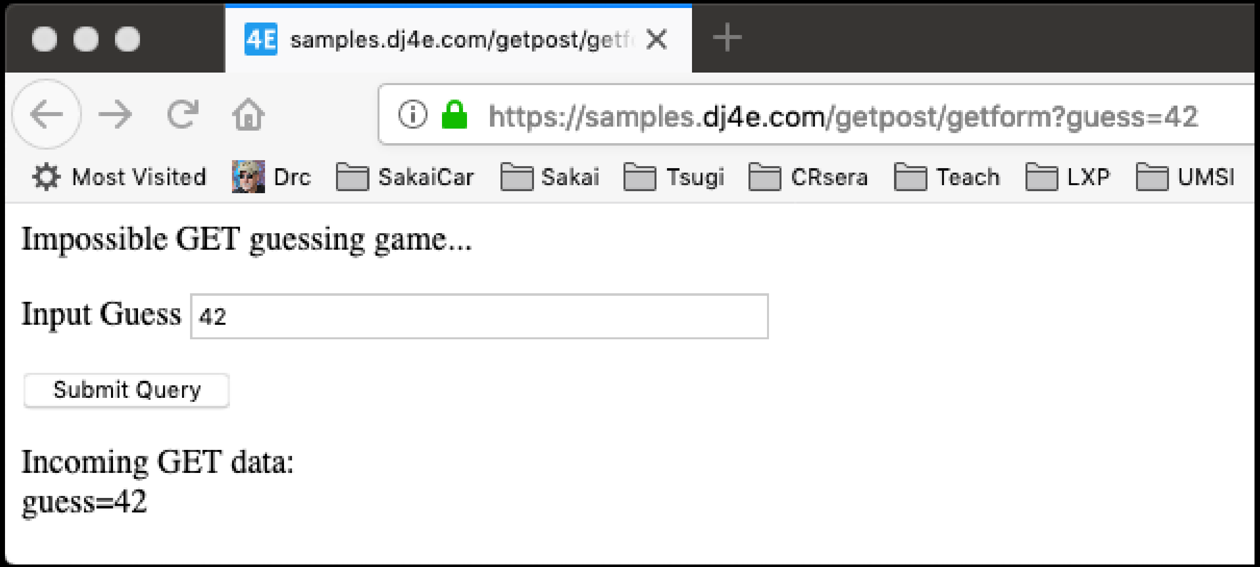
Utility Code – Dump a Dictionary

```python
# Call as dumpdata('GET', request.GET)

def dumpdata(place, data) :
    retval = ""
    if len(data) > 0 :
        retval += '<p>Incoming '+place+' data:<br/>\n'
        for key, value in data.items():
            retval += html.escape(key) + '=' + html.escape(value) + '</br>\n'
        retval += '</p>\n'
    return retval
```

dj4e-samples/getpost/views.py

```
def getform(request):
    response = """<p>Impossible GET guessing game...</p>
        <form>
        <p><label for="guess">Input Guess</label>
        <input type="text" name="guess" size="40" id="guess"/></p>
        <input type="submit"/>
        </form>"""

    response += dumpdata('GET', request.GET)
    return HttpResponse(response)
```

dj4e-samples/getpost/views.py


samples.dj4e.com/getpost/getf
https://samples.dj4e.com/getpost/getform?guess=42
Most Visited
Drc
SakaiCar
Sakai
Tsugi
CRsera
Teach
LXP
UMSI

Impossible GET guessing game...

Input Guess 42

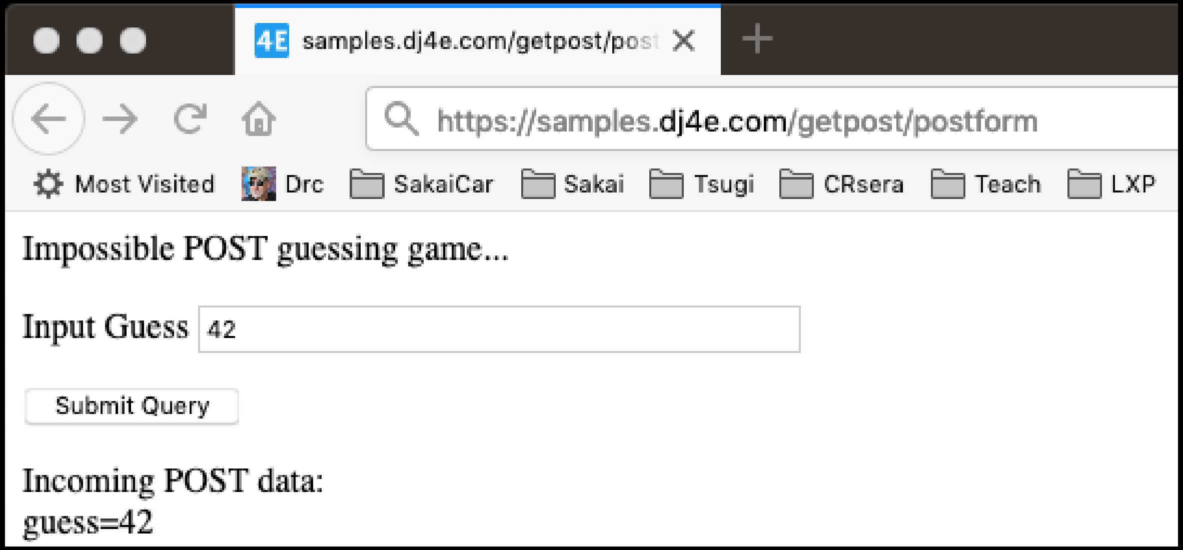
Submit Query

Incoming GET data:
guess=42

https://samples.dj4e.com/getpost/getform

```python
@csrf_exempt
def postform(request):
    response = """<p>Impossible POST guessing game...</p>
        <form method="POST">
        <p><label for="guess">Input Guess</label>
        <input type="text" name="guess" size="40" id="guess"/></p>
        <input type="submit"/>
        </form>"""

    response += dumpdata('POST', request.POST)
    return HttpResponse(response)
```

dj4e-samples/getpost/views.py


samples.dj4e.com/getpost/post
https://samples.dj4e.com/getpost/postform
Most Visited
Drc
SakaiCar
Sakai
Tsugi
CRsera
Teach
LXP

Impossible POST guessing game...

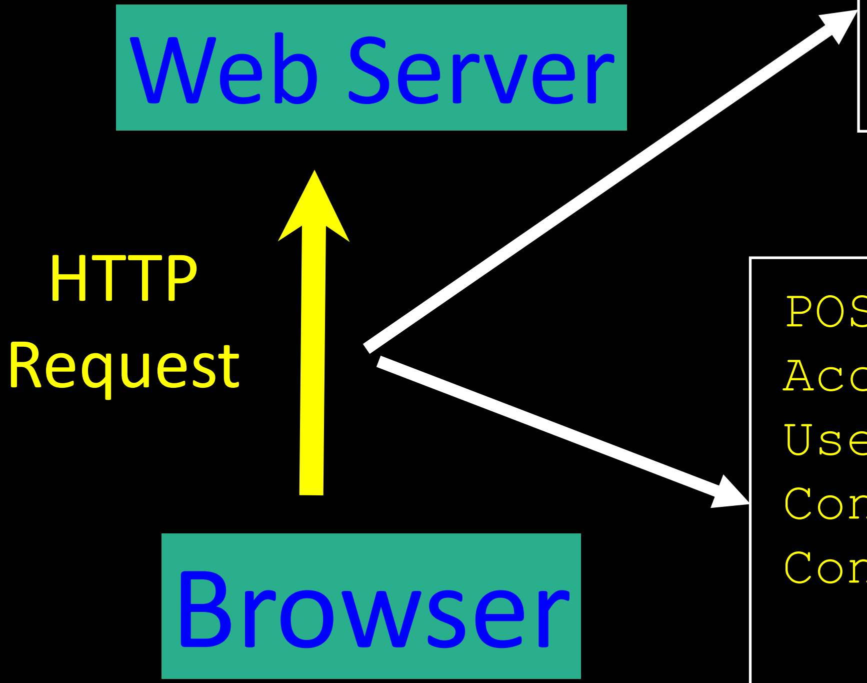
Input Guess 42

Submit Query

Incoming POST data:
guess=42

https://samples.dj4e.com/getpost/postform

Passing Parameters to The Server



```
GET /form/getform?guess=42
Accept: text/html
User-Agent: Lynx/2.4 libwww/2.14
```

```
POST /form/postform
Accept: text/html
User-Agent: Lynx/2.4 libwww/2.14
Content-type: application/x-www-form-urlencoded
Content-length: 13

guess=42
```

```
<input type="text" name="guess" id="yourid" />
```

Rules of the POST/GET Choice

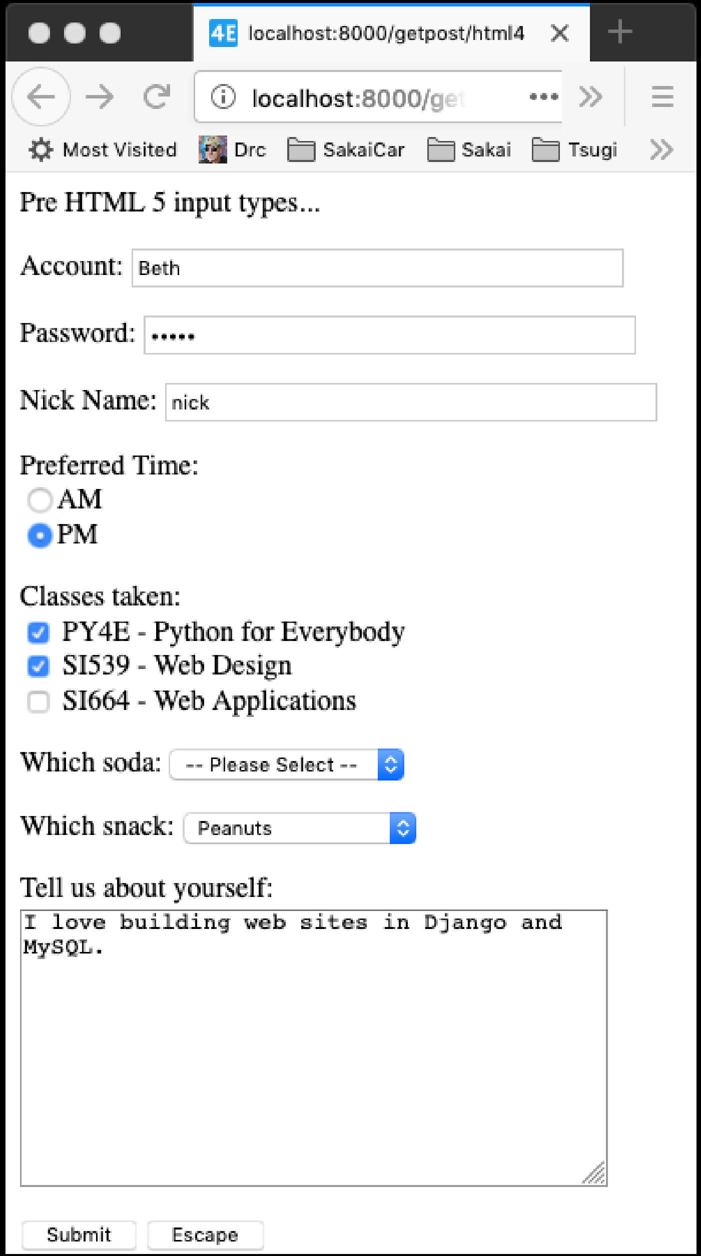
- POST is used when data is being created or modified.
- GET is used when your are reading or searching things.
- GET should never be used to insert, modify or delete data.
- Web search spiders will follow GET URLs but generally not POST URLs.
- GET URLs should be “idempotent” - the same URL should give the “same thing” each time you access it. (i.e. bookmarkable)
- GET has an upper limit of the number of bytes of parameters and values (think about 2K).

FORMS in HTML

Pre HTML5 Input Types

- Text
- Password
- Radio Button
- Check Box
- Select / Drop-Down
- Textarea

```
https://samples.dj4e.com/getpost/html4
dj4e-samples/getpost/templates/getpost/html4.html
```



Pre HTML 5 input types...

Account: Beth

Password: •••••

Nick Name: nick

Preferred Time:
( ) AM
(●) PM

Classes taken:
- [x] PY4E - Python for Everybody
- [x] SI539 - Web Design
- [ ] SI664 - Web Applications

Which soda: -- Please Select --

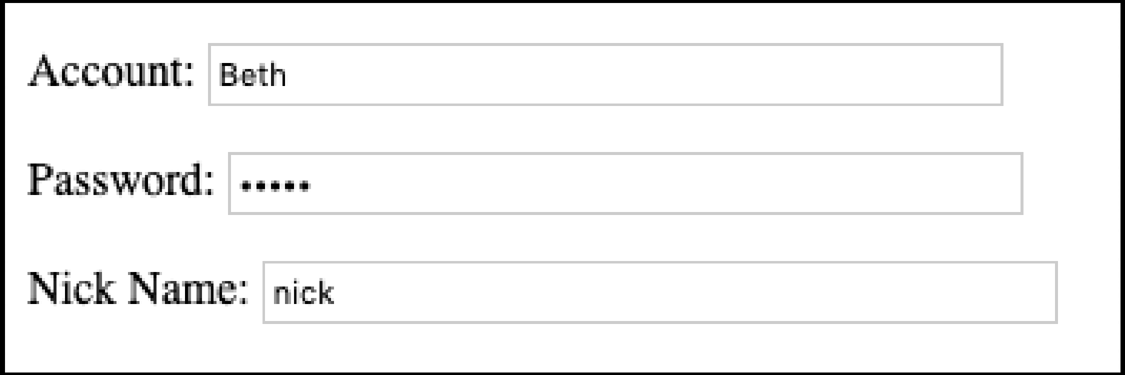
Which snack: Peanuts

Tell us about yourself:

```
I love building web sites in Django and
MySQL.
```

Submit Escape

```
<p>Many field types...</p>
<form method="post">
   <p><label for="inp01">Account:</label>
   <input type="text" name="account" id="inp01" size="40" ></p>
   <p><label for="inp02">Password:</label>
   <input type="password" name="pw" id="inp02" size="40" ></p>
   <p><label for="inp03">Nick Name:</label>
   <input type="text" name="nick" id="inp03" size="40" ></p>
```

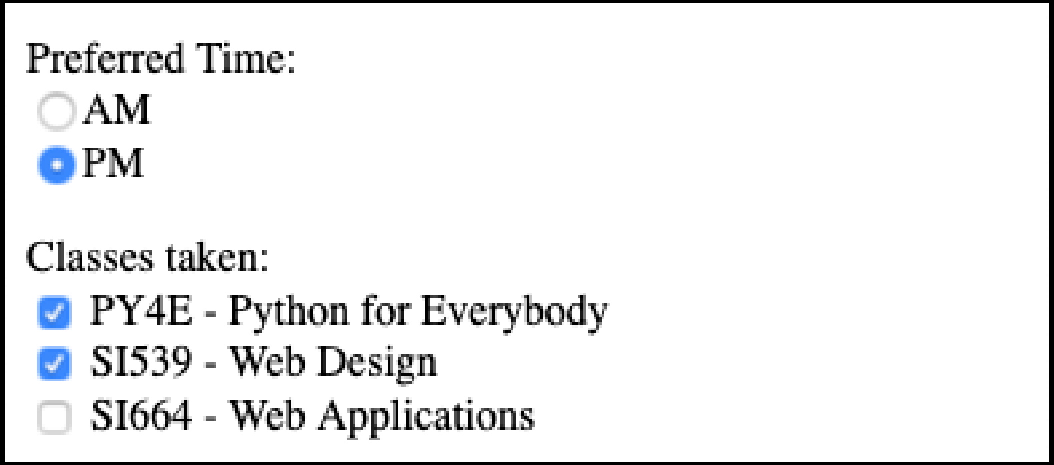


Account:	Beth
Password:	•••••
Nick Name:	nick

```
Incoming POST data:
account=Beth
pw=12345
nick=nick
when=pm
...
```

dj4e-samples/getpost/templates/getpost/html4.html

```
<p>Preferred Time:<br/>
 <input type="radio" name="when" value="am">AM<br>
 <input type="radio" name="when" value="pm" checked>PM</p>
```

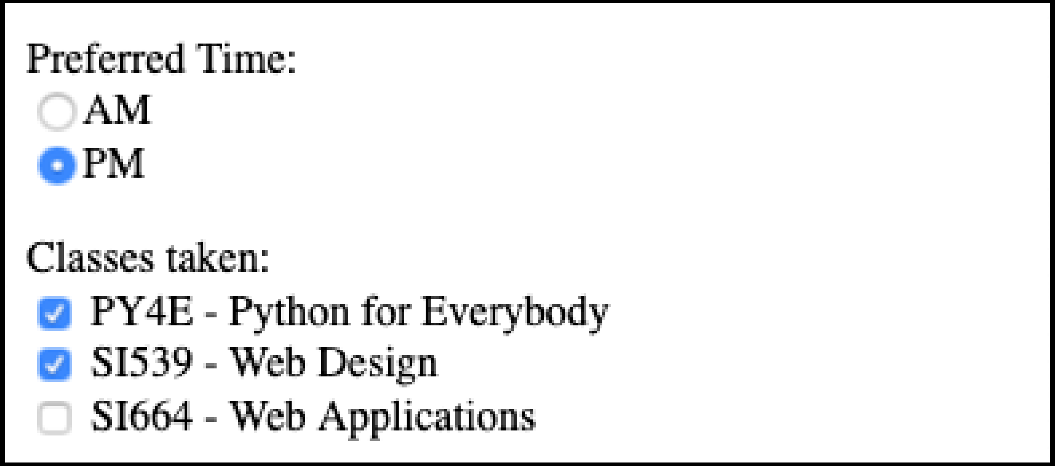


Preferred Time:
- ( ) AM
- (●) PM

Classes taken:
- [✓] PY4E - Python for Everybody
- [✓] SI539 - Web Design
- [ ] SI664 - Web Applications

```
Incoming POST data:
...
when=pm
class1=on
class2=si539
...
```

```
<p>Classes taken:<br/>
    <input type="checkbox" name="class1">
        PY4E - Python for Everybody<br>
    <input type="checkbox" name="class2" value="si539" checked>
        SI539 - Web Design<br>
    <input type="checkbox" name="class3" value="si664">
        SI664 - Web Applications<br>
</p>
```

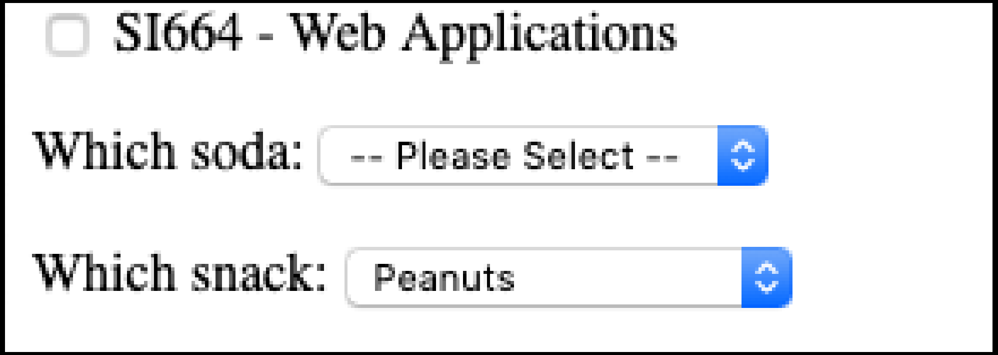

Preferred Time:
○ AM
● PM

Classes taken:
☑ PY4E - Python for Everybody
☑ SI539 - Web Design
☐ SI664 - Web Applications

```
Incoming POST data:
...
when=pm
class1=on
class2=si539
...
```

```
<p><label for="inp06">Which soda:
 <select name="soda" id="inp06">
   <option value="0">-- Please Select --</option>
   <option value="1">Coke</option>
   <option value="2">Pepsi</option>
   <option value="3">Mountain Dew</option>
   <option value="4">Orange Juice</option>
   <option value="5">Lemonade</option>
 </select>
</p>
```

The values can be any string, but numbers are used quite often.

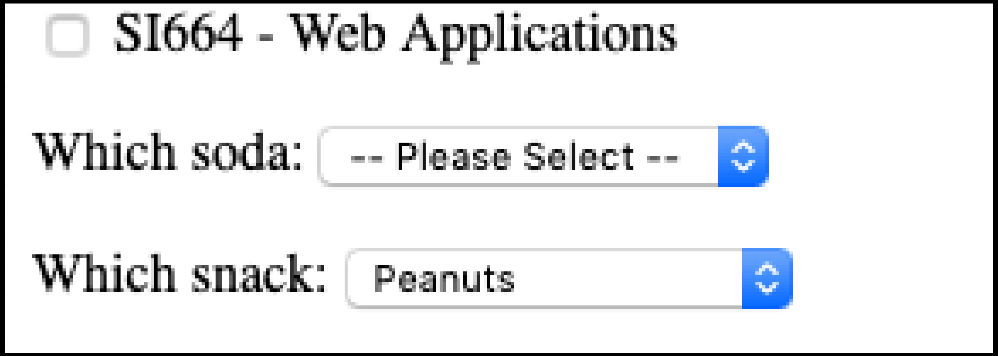

SI664 - Web Applications

Which soda: -- Please Select --

Which snack: Peanuts

```
Incoming POST data:
...
soda=0
snack=peanuts
...
```

```
<p><label for="inp07">Which snack:
  <select name="snack" id="inp07">
    <option value="">-- Please Select --</option>
    <option value="chips">Chips</option>
    <option value="peanuts" selected>Peanuts</option>
    <option value="cookie">Cookie</option>
  </select>
</p>
```


SI664 - Web Applications

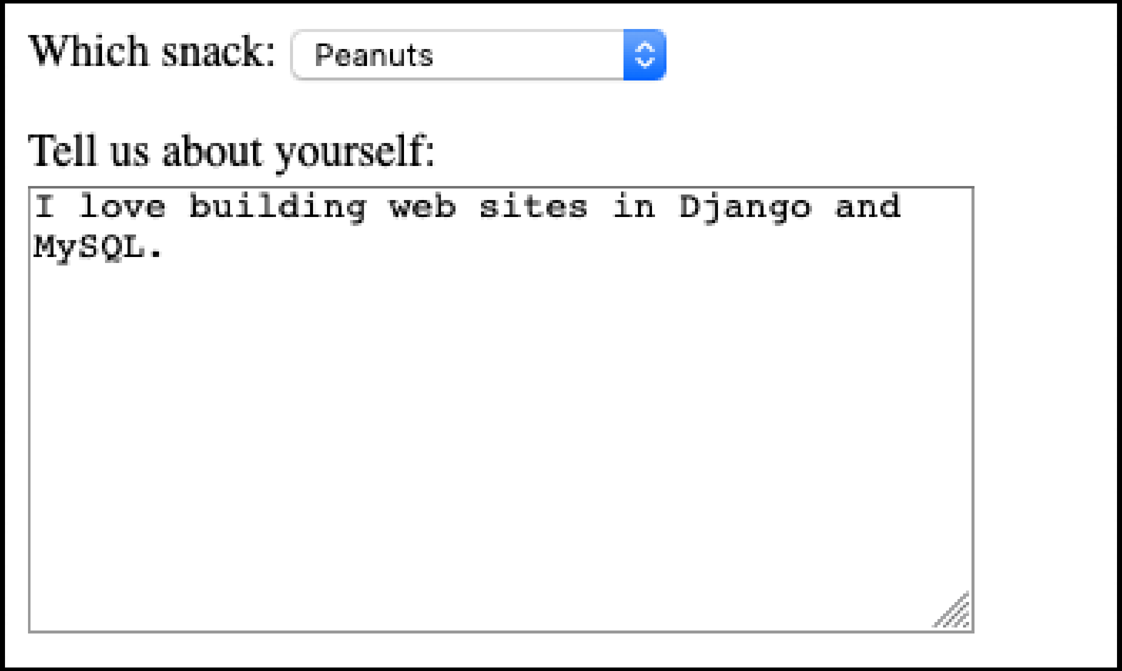
Which soda: -- Please Select --

Which snack: Peanuts

```
Incoming POST data:
...
soda=0
snack=peanuts
...
```

dj4e-samples/getpost/templates/getpost/html4.html

```
<p><label for="inp08">Tell us about yourself:<br/>
  <textarea rows="10" cols="40" id="inp08" name="about">
    I love building web sites in Django and MySQL.
  </textarea>
</p>
```


Which snack: Peanuts

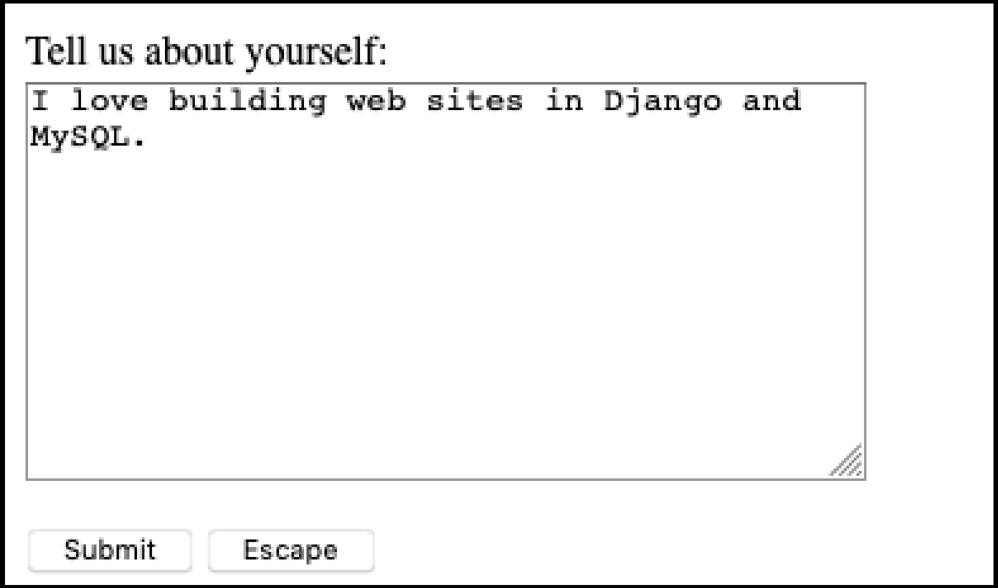
Tell us about yourself:

```
I love building web sites in Django and
MySQL.
```

```
Incoming POST data:
...
snack=peanuts
about=I love building
web sites in Django and
MySQL.
dopost=Submit
```

dj4e-samples/getpost/templates/getpost/html4.html

```
<input type="submit" name="dopost" value="Submit"/>
<input type="button"
  onclick="location.href='http://www.dj4e.com/'; return false;"
  value="Escape">
```


Tell us about yourself:

```
I love building web sites in Django and
MySQL.
```

```
Incoming POST data:
...
snack=peanuts
about=I love building
web sites in Django and
MySQL.
dopost=Submit
```

HTML5 Input Types

- HTML5 defined new input types
- Not all browsers support all input types
- They fall back to type="text"

`https://samples.dj4e.com/getpost/html5`

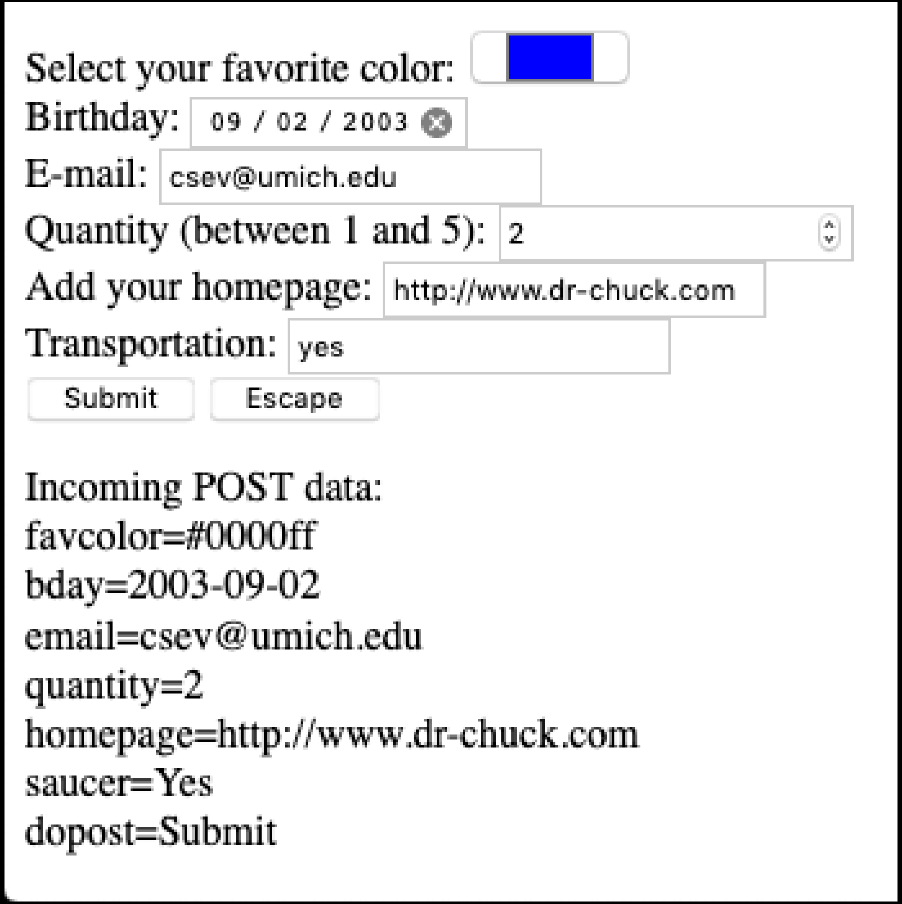
`dj4e-samples/getpost/templates/getpost/html5.html`

`http://www.w3schools.com/html/html5_form_input_types.asp`

```
Select your favorite color:
<input type="color" name="favcolor" value="#0000ff"><br/>
Birthday:
<input type="date" name="bday" value="2003-09-02"><br/>
E-mail:
<input type="email" name="email"><br/>
Quantity (between 1 and 5):
<input type="number" name="quantity"
   min="1" max="5"><br/>
Add your homepage:
<input type="url" name="homepage"><br>
Transportation:
<input type="flying" name="saucer"><br>
```

In-browser validation happens when you press submit.

https://samples.dj4e.com/getpost/html5



Select your favorite color:	🟦
Birthday:	09 / 02 / 2003
E-mail:	csev@umich.edu
Quantity (between 1 and 5):	2
Add your homepage:	http://www.dr-chuck.com
Transportation:	yes

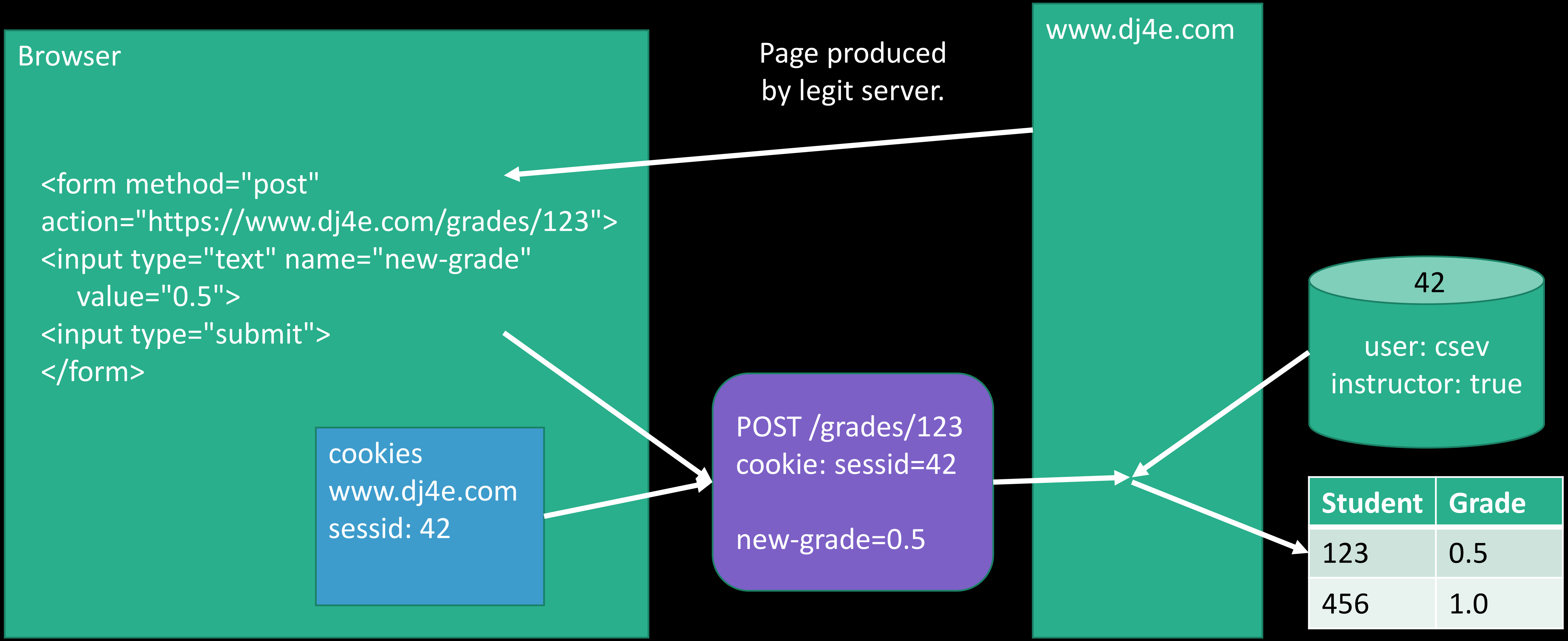
```
Incoming POST data:
favcolor=#0000ff
bday=2003-09-02
email=csev@umich.edu
quantity=2
homepage=http://www.dr-chuck.com
saucer=Yes
dopost=Submit
```

Cross-Site-Request-Forgery (CSRF)

Security

https://en.wikipedia.org/wiki/Cross-site_request_forgery

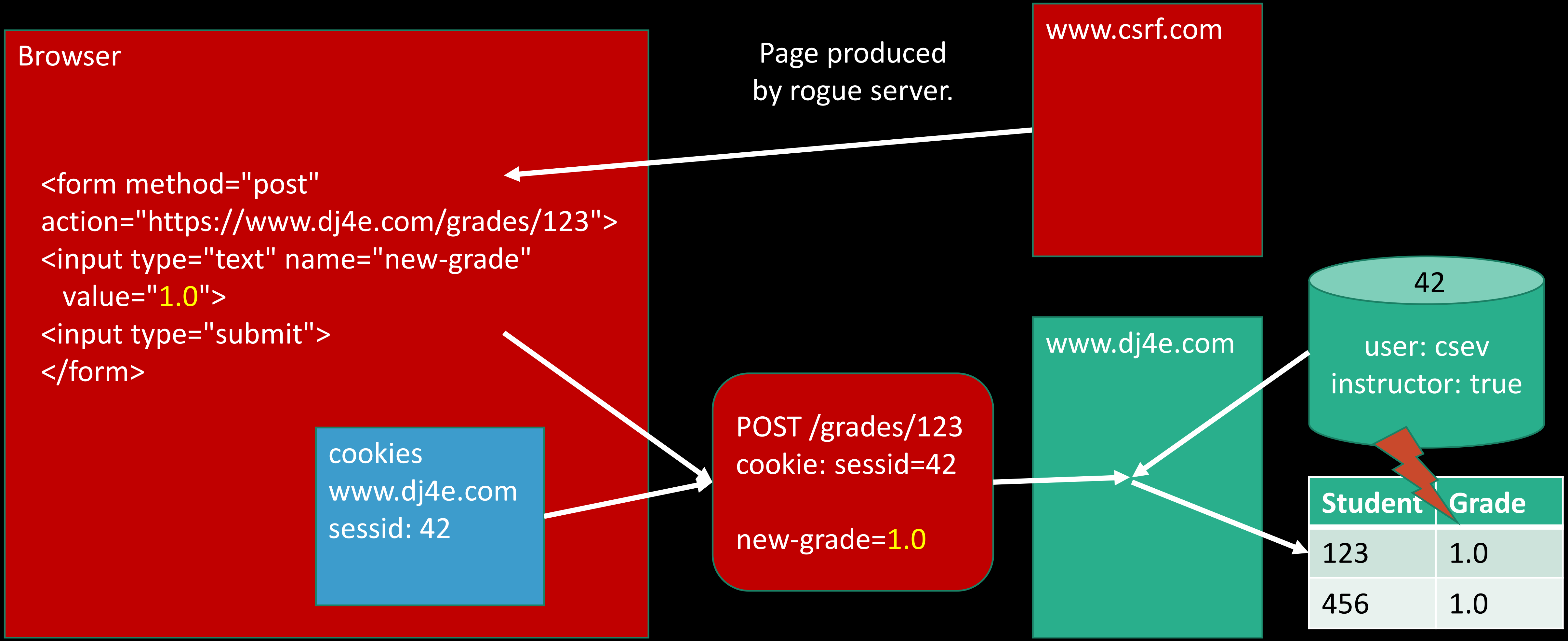
Scenario: Time to Change a Student Grade



```
<form method="post"
action="https://www.dj4e.com/grades/123">
<input type="text" name="new-grade"
    value="0.5">
<input type="submit">
</form>
```

Student	Grade
123	0.5
456	1.0

Attack (without CSRF)

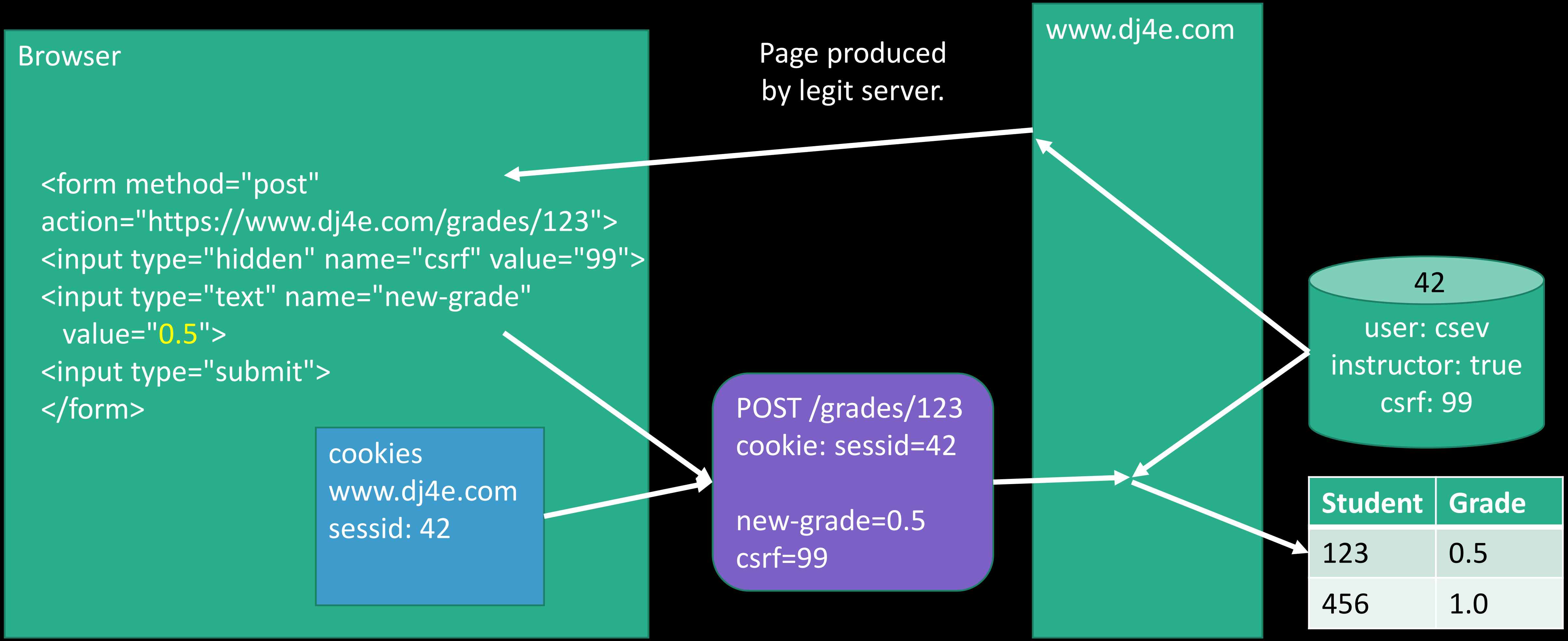


```
<form method="post"
action="https://www.dj4e.com/grades/123">
<input type="text" name="new-grade"
  value="1.0">
<input type="submit">
</form>
```

```
POST /grades/123
cookie: sessid=42

new-grade=1.0
```

Student	Grade
123	1.0
456	1.0

With CSRF

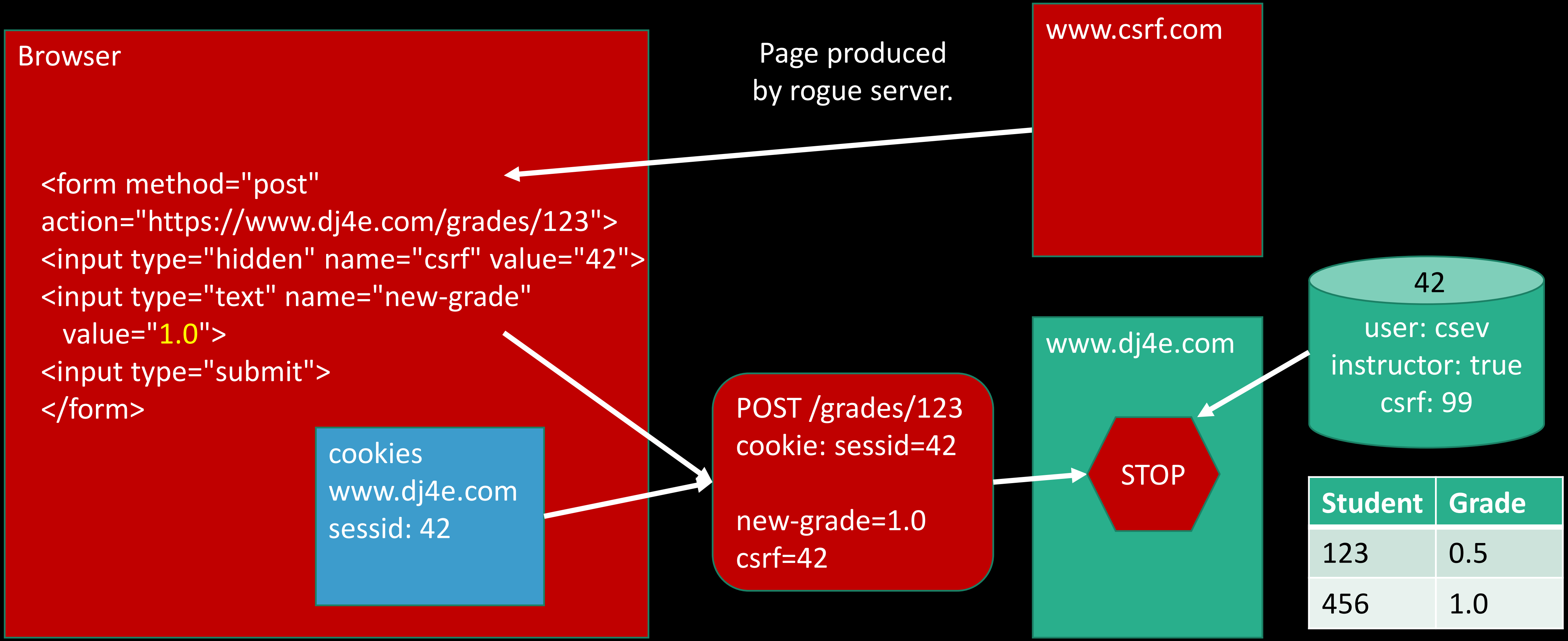


```
<form method="post"
action="https://www.dj4e.com/grades/123">
<input type="hidden" name="csrf" value="99">
<input type="text" name="new-grade"
  value="0.5">
<input type="submit">
</form>
```

```
POST /grades/123
cookie: sessid=42

new-grade=0.5
csrf=99
```

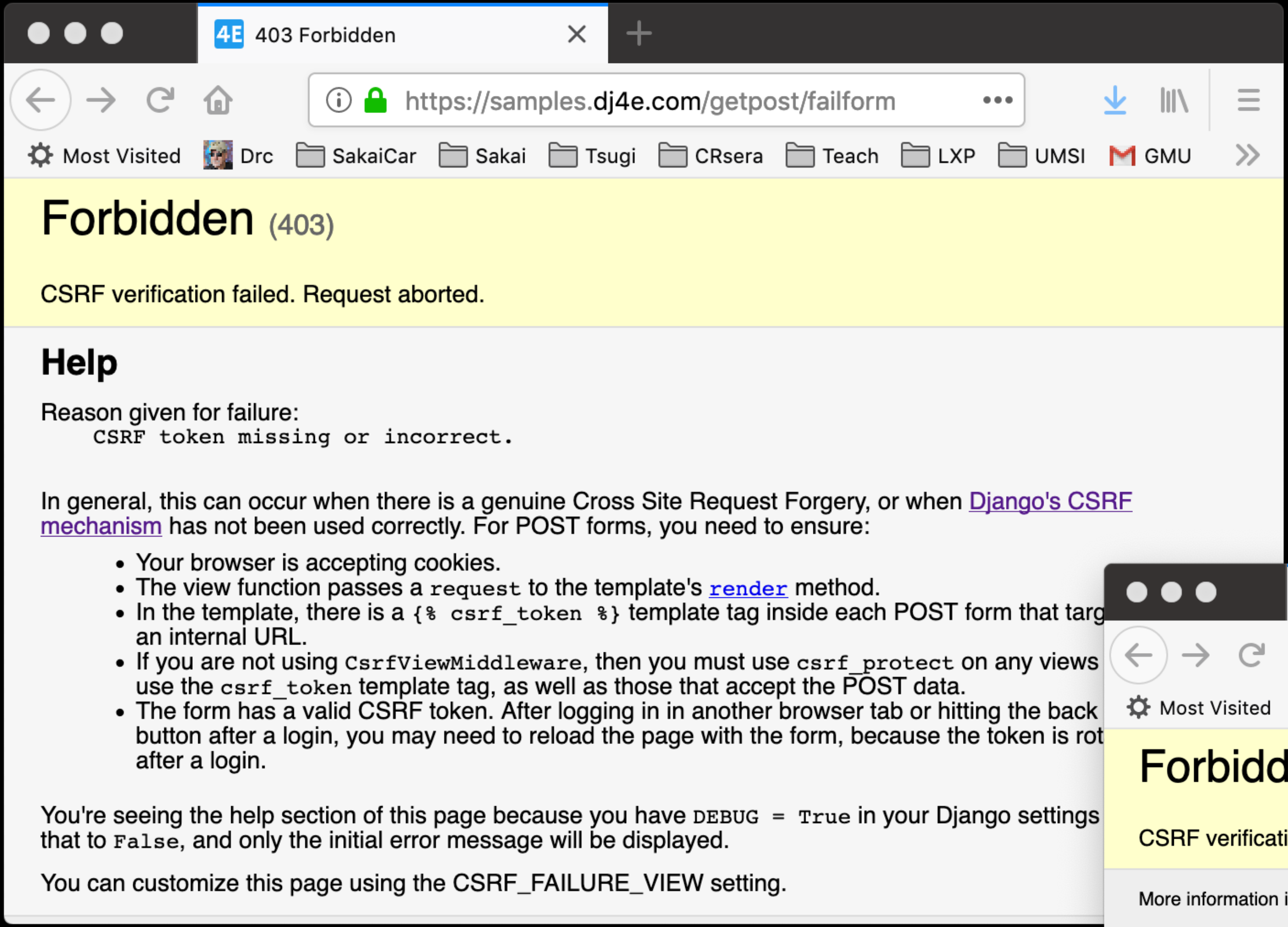
Student	Grade
123	0.5
456	1.0

CSRF Attack Blocked



```
<form method="post"
action="https://www.dj4e.com/grades/123">
<input type="hidden" name="csrf" value="42">
<input type="text" name="new-grade"
  value="1.0">
<input type="submit">
</form>
```

Student	Grade
123	0.5
456	1.0



Forbidden (403)

CSRF verification failed. Request aborted.

Help

Reason given for failure:

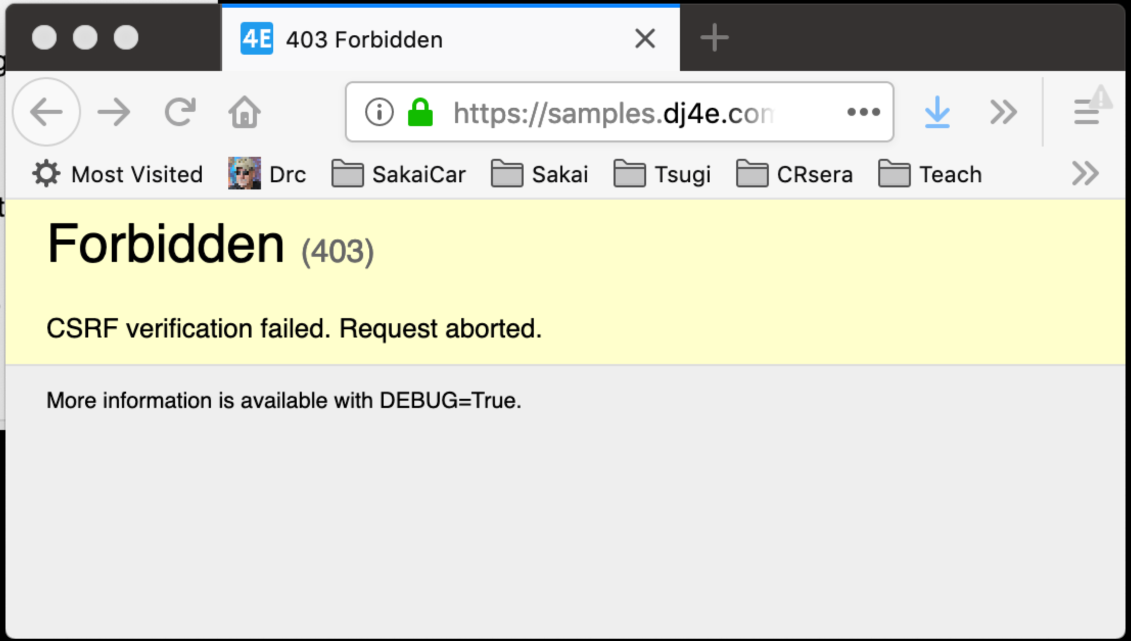
```
    CSRF token missing or incorrect.
```

In general, this can occur when there is a genuine Cross Site Request Forgery, or when Django's CSRF mechanism has not been used correctly. For POST forms, you need to ensure:

- Your browser is accepting cookies.
- The view function passes a `request` to the template's `render` method.
- In the template, there is a `{% csrf_token %}` template tag inside each POST form that targ
an internal URL.
- If you are not using `CsrfViewMiddleware`, then you must use `csrf_protect` on any views
use the `csrf_token` template tag, as well as those that accept the POST data.
- The form has a valid CSRF token. After logging in in another browser tab or hitting the back
button after a login, you may need to reload the page with the form, because the token is rot
after a login.

You're seeing the help section of this page because you have `DEBUG = True` in your Django settings
that to `False`, and only the initial error message will be displayed.

You can customize this page using the CSRF_FAILURE_VIEW setting.



Forbidden (403)

CSRF verification failed. Request aborted.

More information is available with DEBUG=True.

Enabling CSRF defense in Django

- Django has built in support to generate, use, and check CSRF Tokens
- Activated by default in settings.py

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

CSRF in forms

Django CSRF in Templates

dj4e-samples/getpost/templates/getpost/guess.html

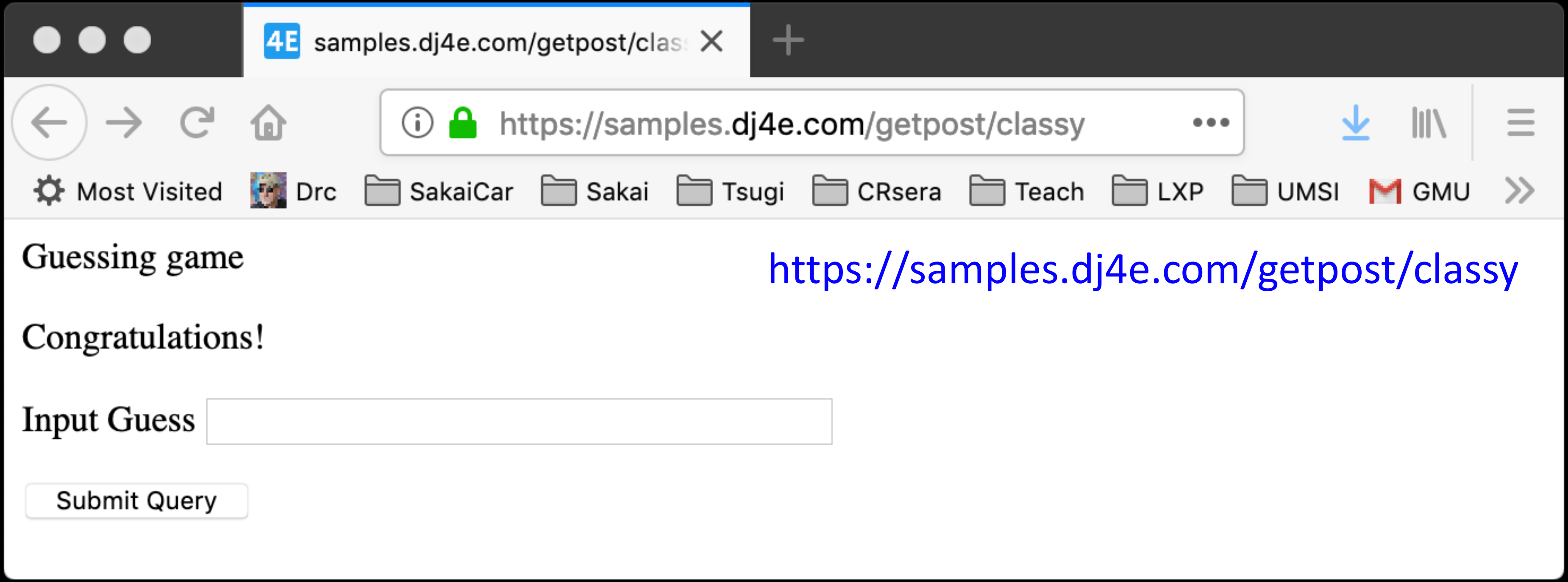
```
<p>Guessing game</p>
{% if message %}
<p>{{ message }}</p>
{% endif %}
<form method="post">
<p><label for="guess">Input Guess</label>
{% csrf_token %}
<input type="text" name="guess" size="40" id="guess"/></p>
<input type="submit"/>
</form>
```

Prática 07

- Criando templates com formulários para escrita dos dados.

POST-Refresh … Oops!

Remember this?


samples.dj4e.com/getpost/clas
https://samples.dj4e.com/getpost/classy
Most Visited
Drc
SakaiCar
Sakai
Tsugi
CRsera
Teach
LXP
UMSI
GMU

Guessing game

Congratulations!

Input Guess

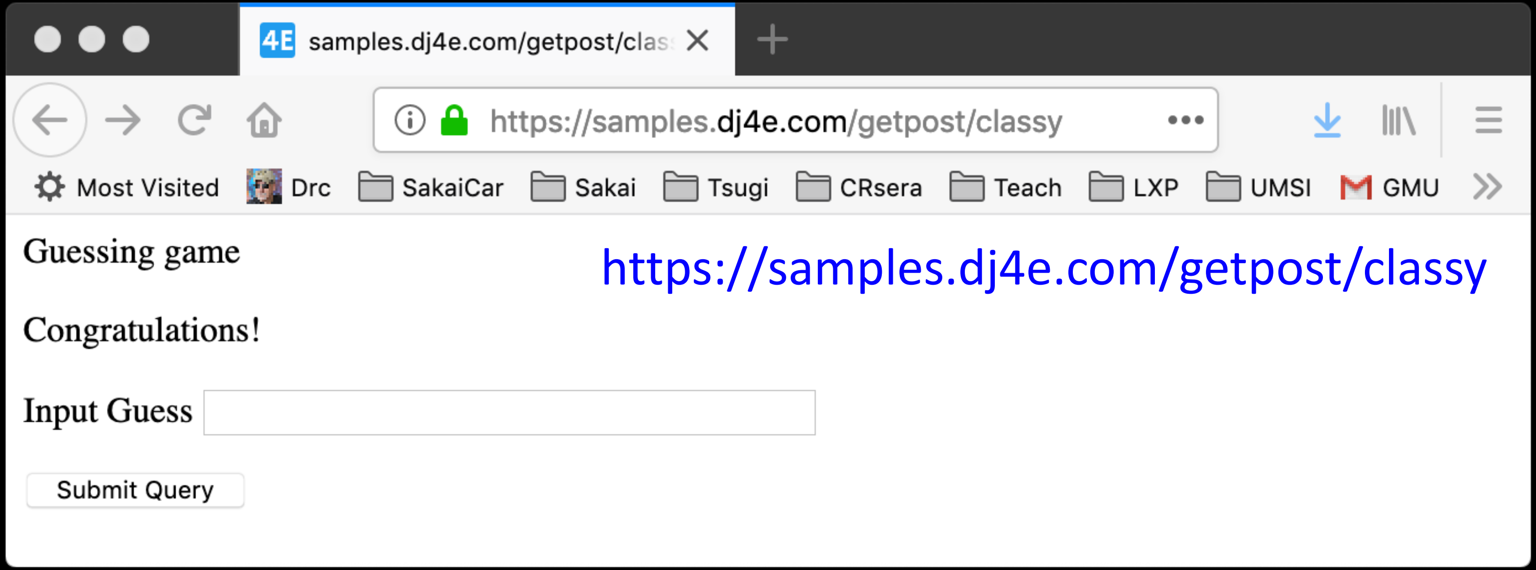
Submit Query

https://samples.dj4e.com/getpost/classy

Success!!!!!

POST / Refresh / ☹

- Once you do a POST and receive 200 status + a page of HTML, if you tell the browser to refresh, the browser will re-send the POST data a second time.
- The user gets a browser pop-up that tries to explain what is about to happen.


samples.dj4e.com/getpost/clas
https://samples.dj4e.com/getpost/classy
Most Visited
Drc
SakaiCar
Sakai
Tsugi
CRsera
Teach
LXP
UMSI
GMU

Guessing game

Congratulations!

Input Guess

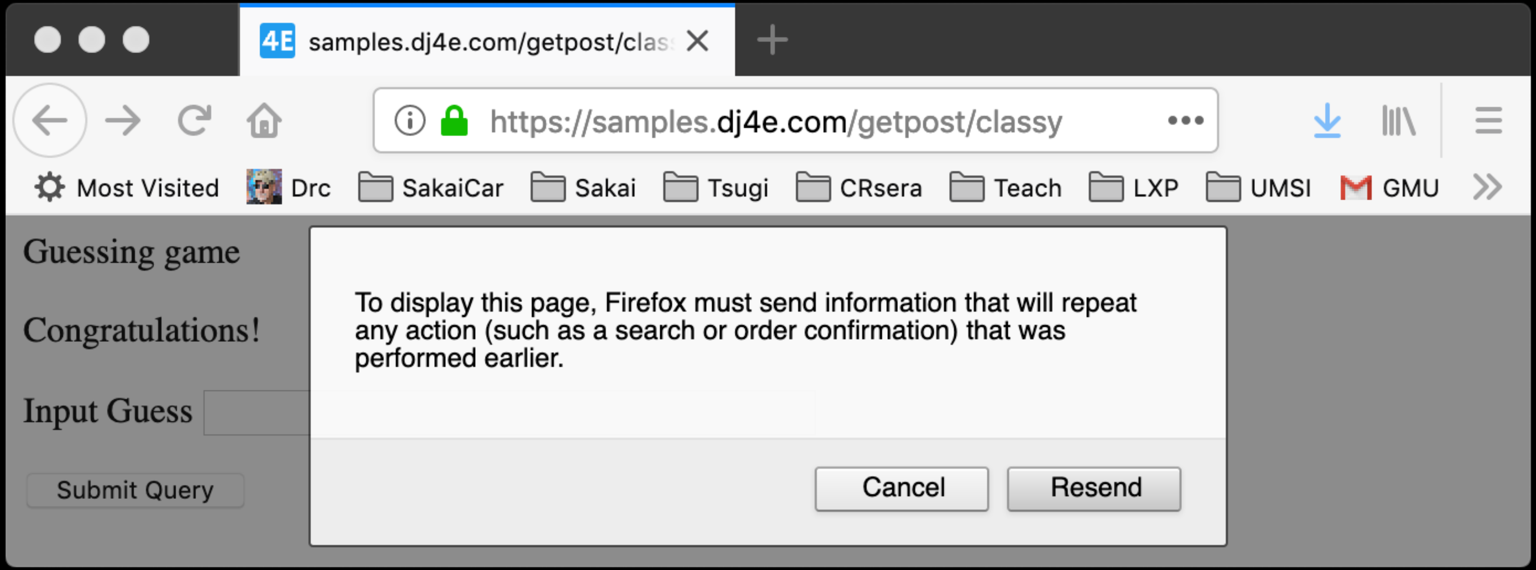
Submit Query

https://samples.dj4e.com/getpost/classy

Make a POST

See Success

Press Refresh


samples.dj4e.com/getpost/clas
https://samples.dj4e.com/getpost/classy
Most Visited
Drc
SakaiCar
Sakai
Tsugi
CRsera
Teach
LXP
UMSI
GMU
Cancel
Resend

Guessing game

Congratulations!

Input Guess

Submit Query

To display this page, Firefox must send information that will repeat any action (such as a search or order confirmation) that was performed earlier.

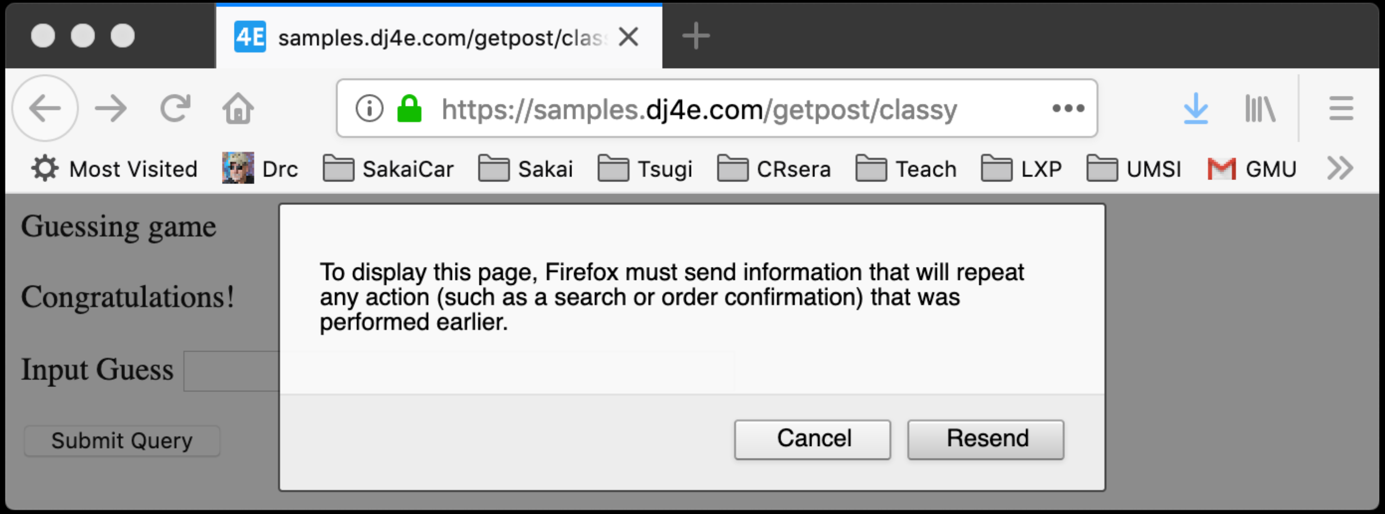
Yucky Message ☹

Don't Allow Double Posts

- Typically POST requests are adding or modifying data whilst GET requests view data
- It may be dangerous to do the same POST twice (say withdrawing funds from a bank account)
- So the browser insists on asking the user (out of your control)
- Kind of an ugly UX / bad usability
- As developers we work so this never can happen

POST-REDIRECT-GET-Refresh

POST Redirect Rule



samples.dj4e.com/getpost/clas

https://samples.dj4e.com/getpost/classy

Guessing game

Congratulations!

Input Guess

To display this page, Firefox must send information that will repeat any action (such as a search or order confirmation) that was performed earlier.

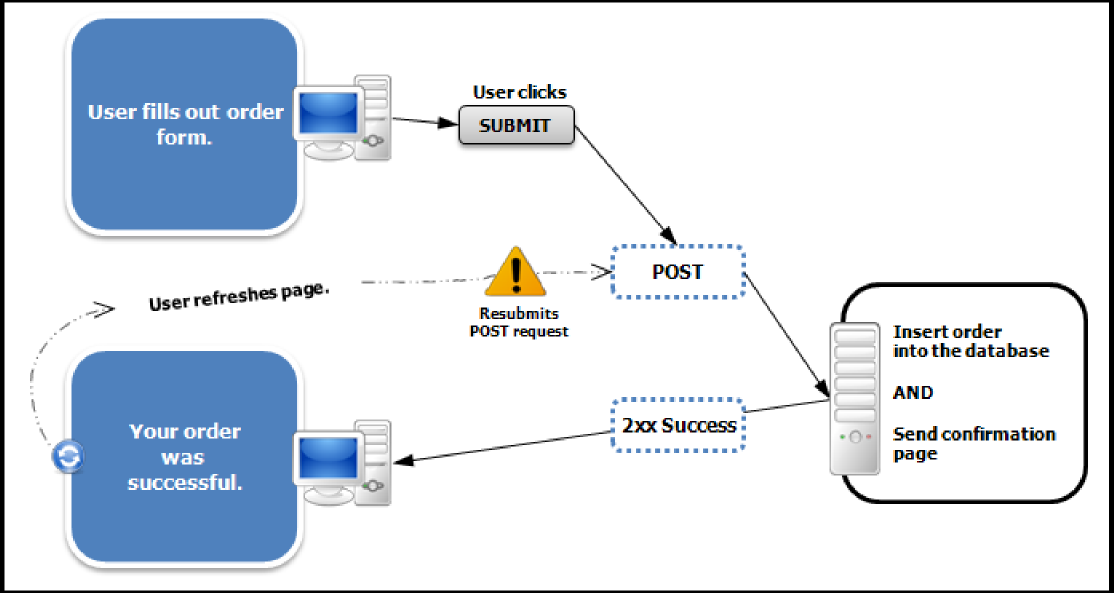
- The simple rule for pages intended for a browser is to never generate a page with HTML content when the app receives POST data and data has been modified
- Must cause a GET by redirecting somewhere - even a GET to the same URL- forcing the browser to make a GET after the POST

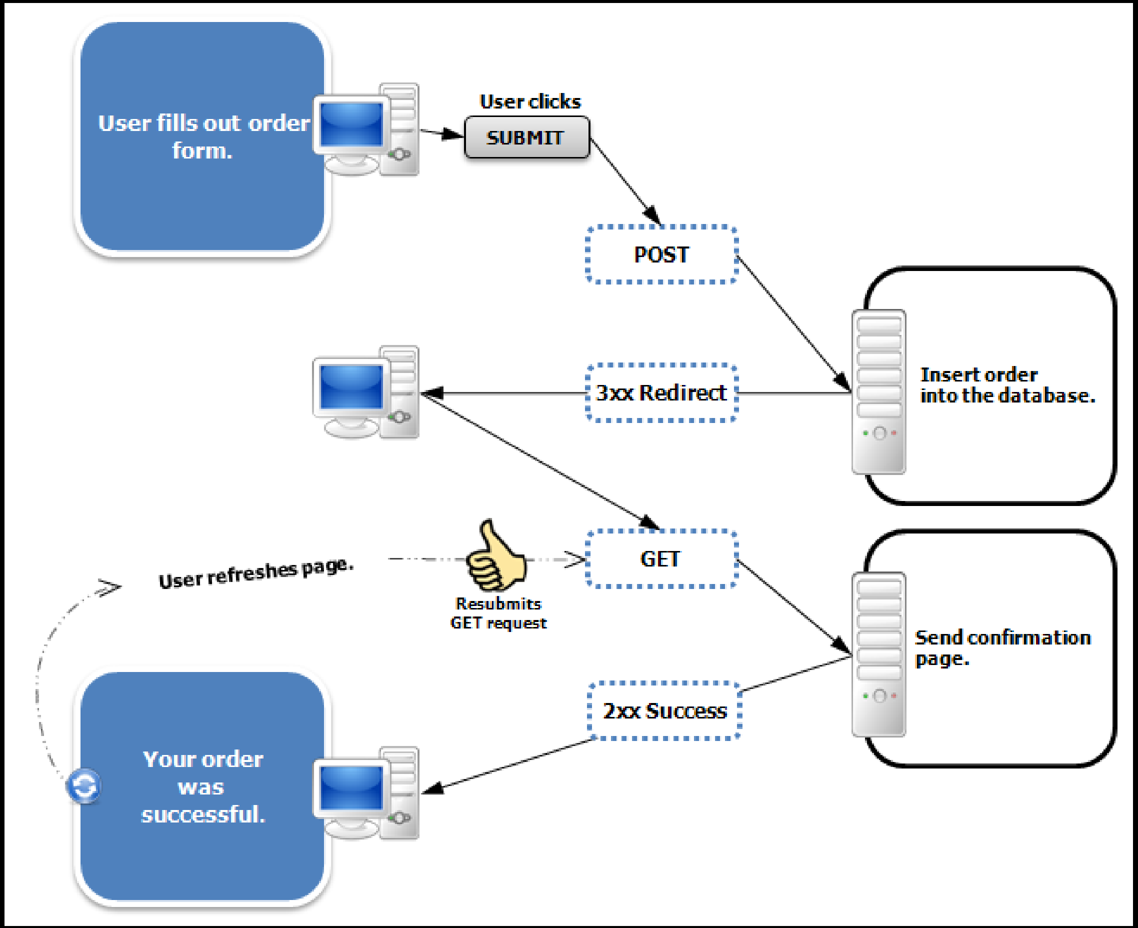
Review: HTTP Status Codes

- http://www.dr-chuck.com/page1.htm - 200 OK
- https://samples.dj4e.com/getpost/failform - 403 Forbidden
    - Post data without CSRF Token
- http://www.wa4e.com/nowhere.htm - 404 Not Found
- http://www.drchuck.com/ - 302 Found / Moved

  Also known as “redirect”

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes





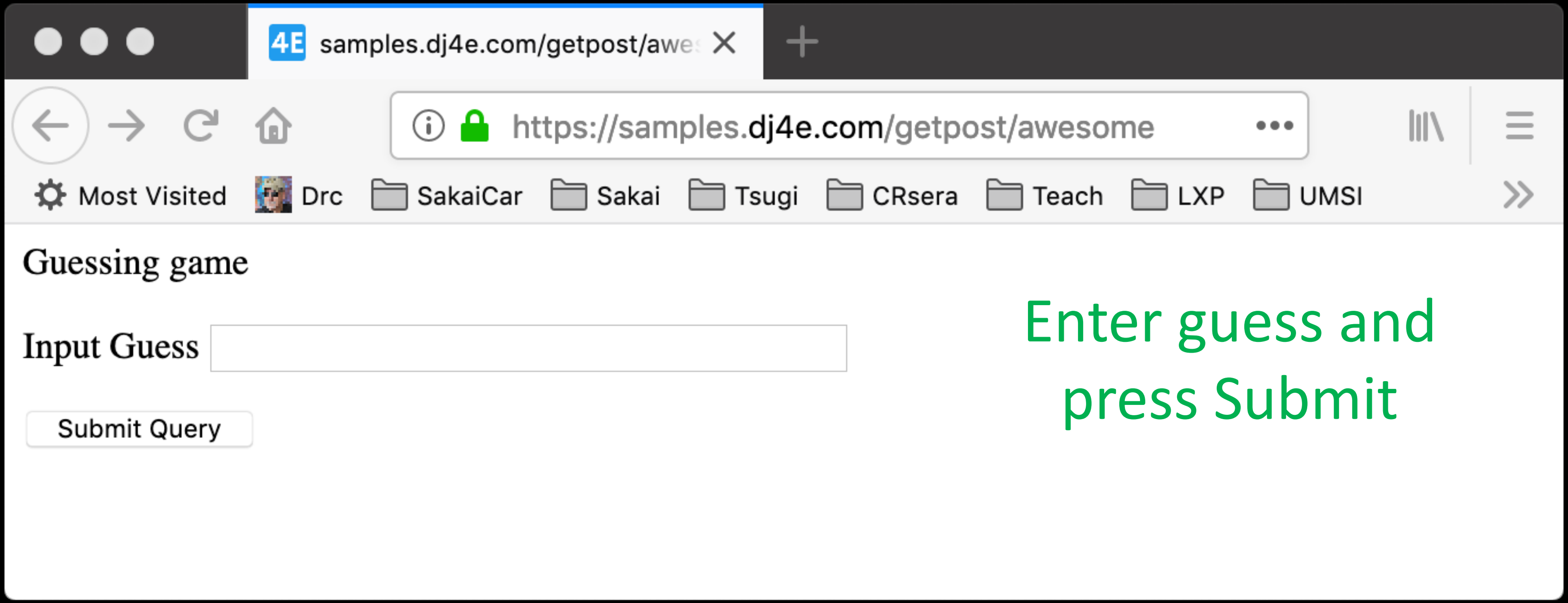
https://en.wikipedia.org/wiki/Post/Redirect/Get dj4e-samples/getpost/views.py

```python
class AwesomeView(View) :
    def get(self, request):
        msg = request.session.get('msg', False)
        if ( msg ) : del(request.session['msg'])
        return render(request, 'getpost/guess.html', {'message' : msg })

    def post(self, request):
        guess = request.POST.get('guess')
        msg = checkguess(guess)
        request.session['msg'] = msg
        return redirect(request.path)
```

dj4e-samples/getpost/templates/getpost/guess.html

```html
<p>Guessing game</p>
{% if message %}
<p>{{ message }}</p>
{% endif %}
<form method="post">
<p><label for="guess">Input Guess</label>
{% csrf_token %}
<input type="text" name="guess" size="40" id="guess"/></p>
<input type="submit"/>
</form>
```

https://samples.dj4e.com/getpost/awesome

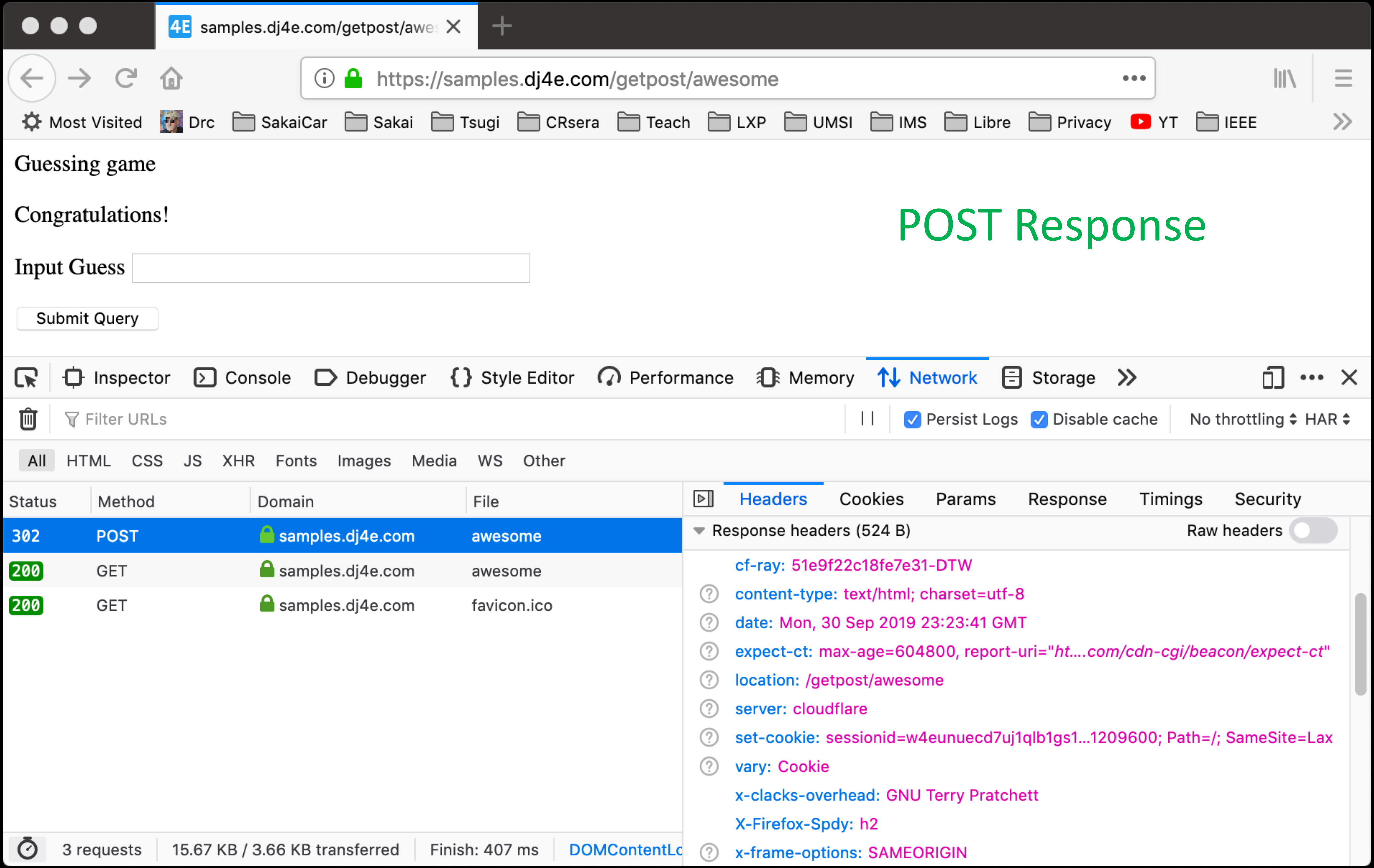

samples.dj4e.com/getpost/awe
https://samples.dj4e.com/getpost/awesome
Most Visited
Drc
SakaiCar
Sakai
Tsugi
CRsera
Teach
LXP
UMSI

Guessing game

Input Guess

Submit Query

Enter guess and press Submit



https://samples.dj4e.com/getpost/awesome

Guessing game

Congratulations!

Input Guess

Submit Query

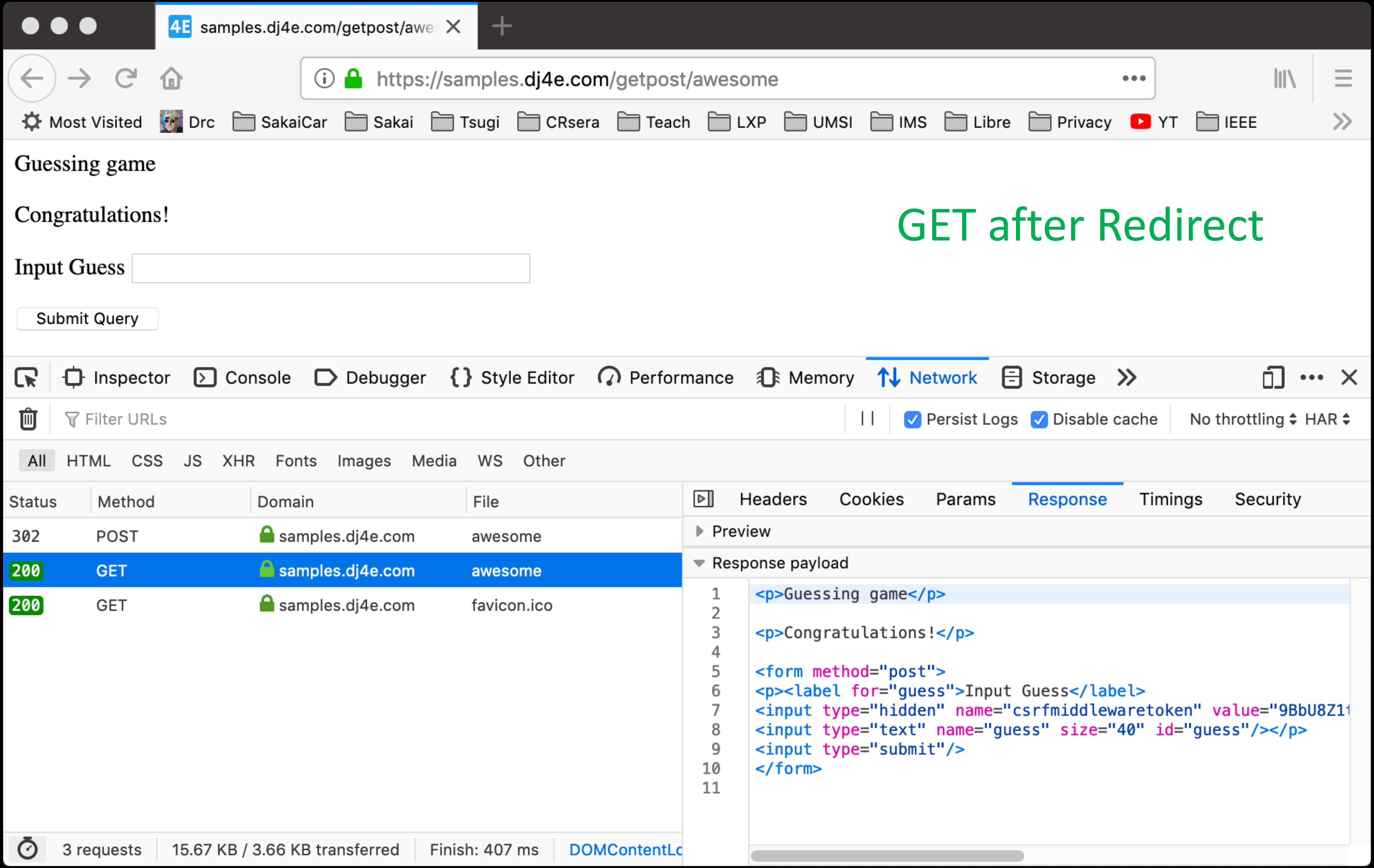
POST Response

Status	Method	Domain	File
302	POST	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	favicon.ico

3 requests | 15.67 KB / 3.66 KB transferred | Finish: 407 ms

Response headers (524 B)

```
cf-ray: 51e9f22c18fe7e31-DTW
content-type: text/html; charset=utf-8
date: Mon, 30 Sep 2019 23:23:41 GMT
expect-ct: max-age=604800, report-uri="ht....com/cdn-cgi/beacon/expect-ct"
location: /getpost/awesome
server: cloudflare
set-cookie: sessionid=w4eunuecd7uj1qlb1gs1...1209600; Path=/; SameSite=Lax
vary: Cookie
x-clacks-overhead: GNU Terry Pratchett
X-Firefox-Spdy: h2
x-frame-options: SAMEORIGIN
```



Guessing game

Congratulations!

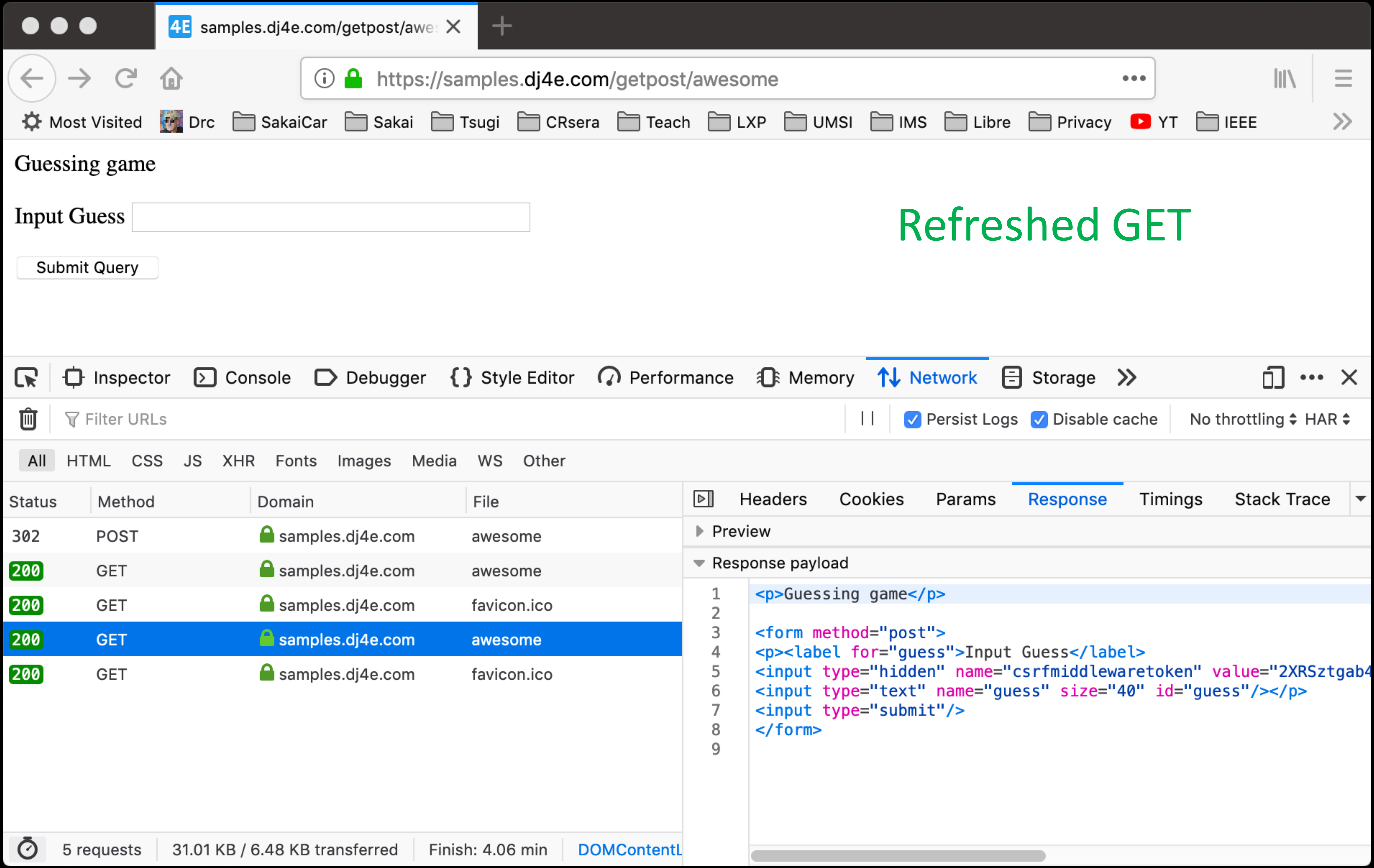
Input Guess

Submit Query

GET after Redirect

Status	Method	Domain	File
302	POST	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	favicon.ico

```
<p>Guessing game</p>

<p>Congratulations!</p>

<form method="post">
<p><label for="guess">Input Guess</label>
<input type="hidden" name="csrfmiddlewaretoken" value="9BbU8Z1
<input type="text" name="guess" size="40" id="guess"/></p>
<input type="submit"/>
</form>
```



Refreshed GET

Guessing game

Input Guess

Submit Query

Status	Method	Domain	File
302	POST	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	favicon.ico
200	GET	samples.dj4e.com	awesome
200	GET	samples.dj4e.com	favicon.ico

```
<p>Guessing game</p>

<form method="post">
<p><label for="guess">Input Guess</label>
<input type="hidden" name="csrfmiddlewaretoken" value="2XRSztgab4
<input type="text" name="guess" size="40" id="guess"/></p>
<input type="submit"/>
</form>
```

5 requests 31.01 KB / 6.48 KB transferred Finish: 4.06 min

The response to a POST must be a redirect

- Pass data to the GET – "flash message pattern"
- Session can be used for flash messages

dj4e-samples/getpost/views.py

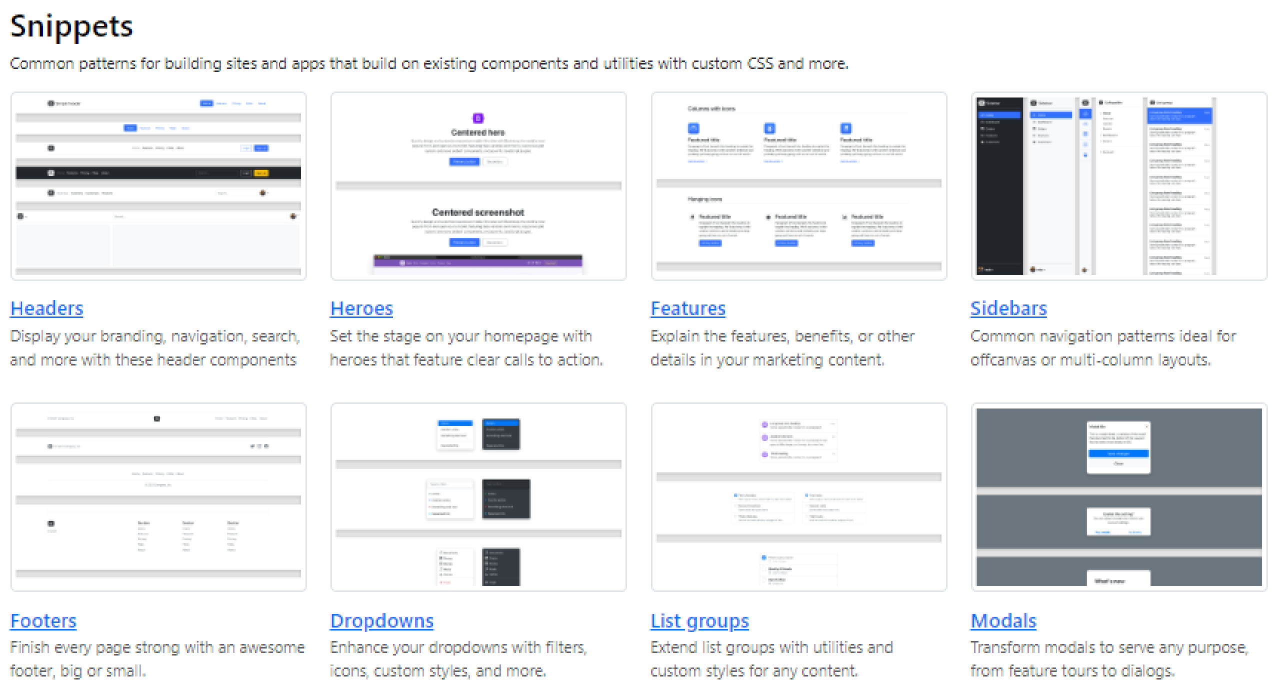
```
class AwesomeView(View) :
    def get(self, request):
        msg = request.session.get('msg', False)
        if ( msg ) : del(request.session['msg'])
        return render(request, 'getpost/guess.html', {'message' : msg })

    def post(self, request):
        guess = request.POST.get('guess')
        msg = checkguess(guess)
        request.session['msg'] = msg
        return redirect(request.path)
```

Summary

- HTML for Forms
- GET versus POST
- CSRF
- POST Redirect GET

Utilizando o Bootstrap

- Adicionando o Bootstrap para suas páginas você pode facilmente alterar a aparência da sua aplicação
- Para isso é necessário utilizar as classes pré-definidas do Bootstrap no atributo *class* de seus elementos HTML.
- Mais informações em https://getbootstrap.com/
  - Em *examples* você consegue achar diversos componentes interessantes



Snippets

Common patterns for building sites and apps that build on existing components and utilities with custom CSS and more.

Headers
Display your branding, navigation, search, and more with these header components

Heroes
Set the stage on your homepage with heroes that feature clear calls to action.

Features
Explain the features, benefits, or other details in your marketing content.

Sidebars
Common navigation patterns ideal for offcanvas or multi-column layouts.

Footers
Finish every page strong with an awesome footer, big or small.

Dropdowns
Enhance your dropdowns with filters, icons, custom styles, and more.

List groups
Extend list groups with utilities and custom styles for any content.

Modals
Transform modals to serve any purpose, from feature tours to dialogs.

Utilizando Bootstrap

- https://www.w3schools.com/django/django_add_bootstrap5.php

- https://www.w3schools.com/bootstrap5/bootstrap_get_started.php

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

White Slides Contributions: Ricardo Costa, CESAR School

Introdução a Django

Ricardo Costa

rac2@cesar.school