Django

Model Template View (MTV)

Examples and source code available on GitHub

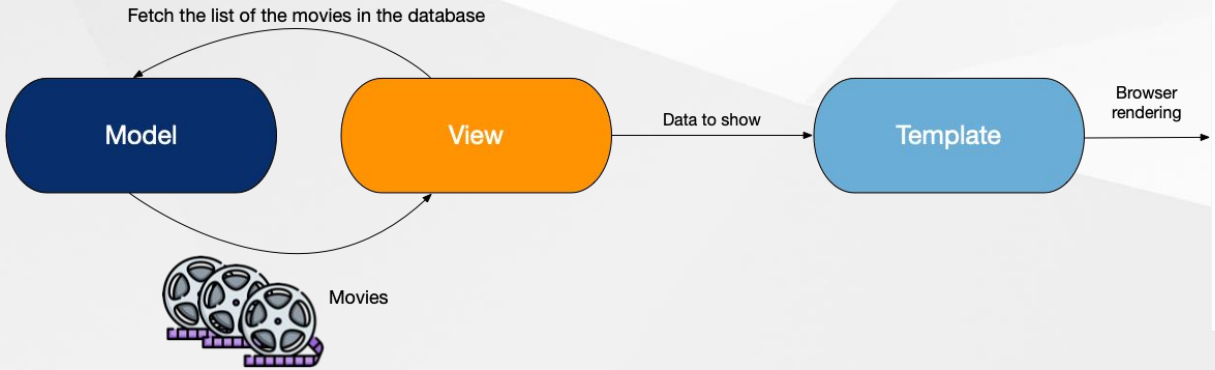
https://github.com/giachell/FIS_21-22

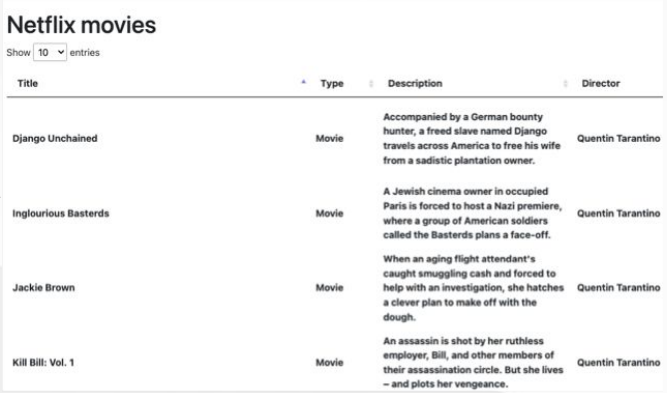
Model Template View (MTV)

https://docs.djangoproject.com/en/3.2/faq/general/#django-appears-to-be-a-mvc-framework

Django is a ***Model Template View (MTV)*** framework, where:

1. ***Model***: define a database prototype for each entity of the data to model.
2. ***Template***: define the presentation layer for the data to show in the browser interface.
3. ***View***: define which data will be presented as a ***Template*** in the browser interface.





Netflix movies

Show 10 entries

Title	Type	Description	Director
Django Unchained	Movie	Accompanied by a German bounty hunter, a freed slave named Django travels across America to free his wife from a sadistic plantation owner.	Quentin Tarantino
Inglourious Basterds	Movie	A Jewish cinema owner in occupied Paris is forced to host a Nazi premiere, where a group of American soldiers called the Basterds plans a face-off.	Quentin Tarantino
Jackie Brown	Movie	When an aging flight attendant's caught smuggling cash and forced to help with an investigation, she hatches a clever plan to make off with the dough.	Quentin Tarantino
Kill Bill: Vol. 1	Movie	An assassin is shot by her ruthless employer, Bill, and other members of their assassination circle. But she lives – and plots her vengeance.	Quentin Tarantino

Models

https://docs.djangoproject.com/en/3.2/topics/db/models/

File: ***netflixapp/models.py***

This file contains all the ***models*** for the data we want to store in the database. For instance, we consider the *Movie* model (right side).

```
from django.db import models
# Create your models here.

class Movie(models.Model):
    """A data structure for Netflix movies and series"""
    title = models.CharField(max_length=1000, blank=False)
    type = models.CharField(max_length=1000, blank=False)
    description = models.CharField(max_length=1000,
blank=True)
    director = models.CharField(max_length=1000, blank=True)
    country = models.CharField(max_length=1000, blank=True)
    cast = models.CharField(max_length=1000, blank=True)
    date_added = models.DateField(blank=True, null=True)
    release_year = models.IntegerField(blank=True, null=True)
    rating = models.CharField(max_length=1000, blank=True)
    duration = models.CharField(max_length=1000, blank=True)
    listed_in = models.CharField(max_length=1000, blank=True)
```

Views

https://docs.djangoproject.com/en/3.2/topics/http/views/#writing-views

File: ***netflixapp/views.py***

A ***view*** is a function that takes a web request and returns a web response. The response can be whatever is supported from a web browser, including: HTML contents, an HTTP response, or even a file such as a document, or an image.

```
from django.shortcuts import render

# Create your views here.

from django.shortcuts import render
from .models import Movie

def index(request):
    movies = Movie.objects.order_by('title')
    context = {'movies': movies}
    return render(request, 'netflixapp/index.html', context)
```

For instance, the ***index*** view above returns the HTML content of the page ***index.html***

Views

https://docs.djangoproject.com/en/3.2/topics/http/views/#writing-views

A ***view*** defines **which** data will be considered for the browser rendering.

In this example, the ***index*** view fetches the list of movies, sorted by title, and creates a dictionary ***context*** from it. Then, the ***context*** dictionary is sent to the ***index.html*** template, which is rendered in the browser.

```
from django.shortcuts import render

# Create your views here.

from django.shortcuts import render
from .models import Movie

def index(request):
    movies = Movie.objects.order_by('title')
    context = {'movies': movies}
    return render(request, 'netflixapp/index.html', context)
```

Templates

https://docs.djangoproject.com/en/3.2/topics/templates/#module-django.template

A ***template*** defines how the data from a ***view*** are rendered in the browser. it is a convenient way to generate HTML dynamically.

In this example, the ***index.html*** template defines the web page structure and how data are presented. Here, the movies' titles are presented in an unordered list (<ul>) .

netflixapp/templates/netflixapp/index.html

```
<!doctype html>
<html lang="en">
<body>
<h1>Netflix movies</h1>
{% if movies %}
   <ul>
   {% for m in movies %}
       <li>{{ m.title }}</li>
   {% endfor %}
   </ul>
{% else %}
   <p>No movies available.</p>
{% endif %}
</body>
</html>
```

Browser output

Netflix movies

- #Alive
- #AnneFrank - Parallel Stories
- #FriendButMarried
- #FriendButMarried 2
- #Roxy
- #Rucker50
- #Selfie
- #Selfie 69
- #blackAF
- #cats_the_mewvie
- #realityhigh
- '76
- '89
- (T)ERROR
- (Un)Well
- 1 Chance 2 Dance
- 1 Mile to You
- 10 Days in Sun City
- 10 jours en or
- 10,000 B.C.

Templates

https://docs.djangoproject.com/en/3.2/topics/templates/#module-django.template

netflixapp/templates/netflixapp/tabular.html

```
<h1>Netflix movies</h1>
{% if movies %}
   <table id="netflix-movies">
   <thead>
       <tr><th>Title</th><th>Type</th><th>Description</th><th>Director</th><th>Country</th>
       <th>Cast</th><th>Date added</th><th>Release year</th><th>Rating</th><th>Duration</th>
       <th>Listed in</th></tr>
   </thead>
   <tbody>
       {% for m in movies %}
          <tr><td>{{ m.title }}</td><td>{{ m.type }}</td><td>{{ m.description }}</td>
               <td>{{ m.director }}</td><td>{{ m.country }}</td><td>{{ m.cast }}</td>
               <td>{{ m.date_added }}</td><td>{{ m.release_year }}</td><td>{{ m.rating }}</td>
               <td>{{ m.duration }}</td><td>{{ m.listed_in }}</td></tr>
     {% endfor %}
   </tbody>
   </table>
{% else %}
   <p>No movies available.</p>
{% endif %}
```

Templates

https://docs.djangoproject.com/en/3.2/topics/templates/#module-django.template

Browser output

Netflix movies

Show 10 entries

Search: Quentin Tarantino

Title	Type	Description	Director	Country	Cast	Date added	Release year	Rating	Duration	Listed in
Django Unchained	Movie	Accompanied by a German bounty hunter, a freed slave named Django travels across America to free his wife from a sadistic plantation owner.	Quentin Tarantino	United States	Jamie Foxx, Christoph Waltz, Leonardo DiCaprio, Kerry Washington, Samuel L. Jackson, Walton Goggins, Dennis Christopher, James Remar, David Steen, Dana Gourrier, Nichole Galicia, Laura Cayouette, Ato Essandoh, Sammi Rotibi, Escalante Lundy, Don Johnson	July 24, 2021	2012	R	165 min	Action & Adventure, Dramas
Inglourious Basterds	Movie	A Jewish cinema owner in occupied Paris is forced to host a Nazi premiere, where a group of American soldiers called the Basterds plans a face-off.	Quentin Tarantino	Germany, United States	Brad Pitt, Mélanie Laurent, Christoph Waltz, Eli Roth, Michael Fassbender, Diane Kruger, Daniel Brühl, Til Schweiger, Gedeon Burkhard, Jacky Ido, B.J. Novak, Sylvester Groth, Martin Wuttke	July 22, 2019	2009	R	153 min	Action & Adventure

An introduction to

HTML

HyperText Markup Language

HTML Introduction

What is HTML?

- HTML stands for **HyperText Markup Language**. The HyperText part refers to the fact that HTML allows you to create links that allow visitors to move from one page to another quickly and easily.
- A markup language allows you to annotate text, and these annotations provide additional meaning to the contents of a document.
- HTML allows to **structure web pages** so that their information content is arranged according to the layout specified by HTML **tags**.

<u>How to view the HTML source code of a web page using a browser</u>:

Right-click in an HTML page and click the "*View Page Source*" option (in Chrome) or "*View Source*" in Edge. A similar option is present also in other browsers. There is also a keyboard shortcut **CTRL + U** (valid for Chrome/Firefox on Windows and Linux).

Reference:
https://www.w3schools.com/html/

A simple HTML page

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default

Run »

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>

<p>This is a paragraph</p>

</body>
</html>
```

This is a Heading

This is a paragraph.

11

W3schools try-it website: https://www.w3schools.com/html/tryit.asp

A simple HTML page

Run

Result Size: 561 x 410

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>

<p>This is a paragraph.</p>

<a href="https://elearning.unipd.it/math/course/view.php?id=781">Fundamentals of Information Systems</a>

</body>
</html>
```

This is a Heading

This is a paragraph.

Fundamentals of Information Systems

W3schools try-it website: https://www.w3schools.com/html/tryit.asp

The anatomy of an HTML tag

Example 1: the paragraph tag

 This is a paragraph

Opening tag The actual text of the paragraph **Closing tag**

p is the **Tag name** and stands for paragraph

The anatomy of an HTML tag

Example 2: the heading tag

<h1> This is a Heading </h1>

Opening tag The actual text of the heading **Closing tag**

h1 is the **Tag name** and stands for first-level heading

The anatomy of an HTML tag

Example 3: the hyperlink tag

```
<a href="https://elearning.unipd.it/math/course/view.php?id=781">FIS 21-22</a>
```

the **href** attribute indicates the link's destination

a is the **Tag name** and defines a **hyperlink**

The actual text of the link.

NB: This is the only part visible to the user.

The anatomy of an HTML web page

```
<html>
  <body>
    <h1>This is the Main Heading</h1>
    <p>This text might be an introduction to the
    rest of the page. And if the page is a long one
    it might be split up into several sub
    headings.<p>
    <h2>This is a Sub-Heading</h2>
    <p>Many long articles have sub-headings so to
    help you follow the structure of what is being
    written. There may even be sub-sub-headings (or
    lower-level headings).<p>
  </body>
</html>
```

The **html** tag is the root element and contains all the elements that make up a web page.

The **body** tag contains all the elements that compose the web page view.

Other relevant HTML tags

- **HTML Headings**: defined with the <h1> to <h6> tags. <h1> defines the most important heading (biggest); <h6> defines the least important heading (smallest).

- **HTML Images**: defined with the <img> tag.

  Properties:

  - The src attribute allows to specify the image source file.
  - The alt attribute allows to specify the alternative text shown in case the image cannot be displayed.
  - The width and height attributes specify the width and the height respectively of the image.

- **HTML line break**: The <br> tag inserts a single line break. After the line break the text is wrapped.

Reference: https://www.w3schools.com/html/html_basic.asp

Other relevant HTML tags

- **HTML Lists**: there are two main types of HTML lists:
  1. **Ordered lists**: defined with tag
  2. **Unordered lists**: defined with tag

  Each list item starts with the <li> tag
- **HTML tables**: defined using the <table> tag.

  Properties:

  - Each table row is defined with a <tr> tag. Each table header is defined with a <th> tag. Each table cell is defined with a <td> tag.
  - By default, the text in <th> elements are bold and centered.
  - By default, the text in <td> elements are regular and left-aligned.

Reference: https://www.w3schools.com/html/html_basic.asp

HTML forms

HTML forms allow the users to provide information by filling up input fields (e.g. text boxes, check boxes and drop-down lists).

The HTML <form> define an HTML form and act as a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

Properties:

- The **action** attribute allows to specify the URL of the destination web page that will receive the form data. (Where the form data are sent)
- The **method** attribute allows to specify the HTTP method (e.g. GET, POST) used to send the form data. (How the form data are sent)

Reference:

1. https://www.w3schools.com/html/html_forms.asp
2. https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_submit

HTML form example

Run »

Result Size: 476 x 482

```html
<!DOCTYPE html>
<html>
<body>
<h2>HTML Forms</h2>

<form action="/action_page.php">

  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">

</form>

<p>If you click the "Submit" button, the form-data will be sent to
a page called "/action_page.php".</p>

</body>
</html>
```

HTML Forms

First name:

John

Last name:

Doe

Submit

If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".

Reference:

- https://www.w3schools.com/html/html_forms.asp
- https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_submit

HTML form example

Run

Result Size: 476 x 482

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Forms</h2>

<form action="/action_page.php">

  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" valu
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value= oe"><br><br>
  <input type="submit" value="Submit">

</form>

<p>If you click the "Submit" button, the form-data will be sent to
a page called "/action_page.php".</p>

</body>
</html>
```

How we can handle the form data sent?

Submit

If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".

Reference:

- https://www.w3schools.com/html/html_forms.asp
- https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_submit

Handle form data

https://docs.djangoproject.com/en/3.2/topics/forms/#working-with-forms

We can handle form data in our Django ***views***.

Goal: we want to create an interface for *Create, Read, Update, and Delete (CRUD)* operations on our data.

Note: We have already created a web page for reading the movies' data. Hence, we focus now on *Create, Update, and Delete* operations.

Handle form data

https://docs.djangoproject.com/en/3.2/topics/forms/#working-with-forms

CREATE: we want to add a movie to our catalog.

Netflix movies

Add a new movie to the catalog

title: Gladiator
type: Movie
description: In 180 AD, Hispano-Roma
director: Ridley Scott
country: USA
cast: Russell Crowe Joaquin Ph
date_added: 2021-11-05
release_year: 2000
rating: TV-14
duration: 155 min
listed_in: Historical, Dramatic

Add

Netflix movies

Add a new movie to the catalog

Congratulations: your movie **Gladiator** has been added!

Details:

Title	Type	Description	Director	Country	Cast	Date added	Release year	Rating	Duration	Listed in
Gladiator	Movie	In 180 AD, Hispano-Roman General Maximus Decimus Meridius intends to return to his home after he leads the Roman army to victory against the Germanic tribes near Vindobona on the Limes Germanicus. Emperor Marcus Aurelius tells Maximus that his own son, Commodus, is unfit to rule and that he wishes Maximus to succeed him, as regent, to help save Rome from corruption and restore the Roman Republic. Upon hearing this, Commodus murders his father.	Ridley Scott	USA	Russell Crowe Joaquin Phoenix Connie Nielsen Oliver Reed Derek Jacobi Djimon Hounsou Richard Harris Tommy Flanagan	2021-11-05	2000	TV-14	155 min	Historical, Dramatic

Handle form data

View definition in ***netflixapp/views.py***

```
def add_movie(request):
    message = ""
    context = {"message": message, "message_type": None}
    if request.method == 'POST':
        title = request.POST.get("title", None)
        type = request.POST.get("type", None)
        # other fields omitted for the sake of neatness
        if not title:
            message = "An error occurred, your movie has not been added"
            context = {"message": message, "message_type": "error"}
        else:
            m = Movie(title=title, type=type, description=desc,
                      director=director, country=country, cast=cast,
                      date_added=date_added, duration=duration, rating=rating,
                      listed_in=listed_in, release_year=release_year)
            m.save()
            message = f"Congratulations: your movie <b>{m.title}</b> has been added!"
            context = {"message": message, "movie": m, "message_type": "success"}
    return render(request, 'netflixapp/add-movie.html', context)
```

Handle form data

Template definition in ***netflixapp/templates/netflixapp/add-movie.html***

```
<h1>Netflix movies</h1>
<h3>Add a new movie to the catalog</h3>
{% if movie %}
   <p class="alert alert-success">Congratulations: Your movie '{{ movie.title }}' has been
added!</p>
   <p>Details:</p>
   <table class="table table-bordered" id="movie-details">
   <thead class="thead-light">
       <tr><th>Title</th><th>Type</th><th>Description</th>
           <th>Director</th><th>Country</th><th>Cast</th>
           <th>Date added</th><th>Release year</th><th>Rating</th>
           <th>Duration</th><th>Listed in</th></tr>
   </thead>
   <tbody>
      <tr><td>{{ movie.title }}</td><td>{{ movie.type }}</td><td>{{ movie.description }}</td>
   <td>{{ movie.director }}</td><td>{{ movie.country }}</td><td>{{ movie.cast }}</td>
   <td>{{ movie.date_added }}</td><td>{{ movie.release_year }}</td><td>{{ movie.rating }}</td>
   <td>{{ movie.duration }}</td><td>{{ movie.listed_in }}</td></tr>
   </tbody>
   </table>
```

Handle form data

https://docs.djangoproject.com/en/3.2/topics/forms/#working-with-forms

DELETE: we want to remove a movie from our catalog.

Netflix movies

Show 10 entries

Search: title

Title	Type	Description	Director	Country	Cast	Date added	Release year	Rating	Duration	Listed in	Delete
A dummy title						None	None				
A Witches' Ball	Movie	Beatrix can't wait to be inducted as a witch, but an unfortunate incident threatens to take her pending title away if she doesn't act fast.	Justin G. Dyck	Canada	Morgan Neundorf, Karen Slater, Loukia Ioannou, Will Ennis, Renee Stein, Keith Cooper, Paul Mason, Lisa Scenna, Ashley Rogan	Oct. 1, 2018	2017	PG	91 min	Children & Family Movies	
Article 15	Movie	The grim realities of caste discrimination come to light as an entitled but upright city cop ventures into India's heartland to investigate a murder.	Anubhav Sinha	India	Ayushmann Khurrana, Nassar, Manoj Pahwa, Kumud Mishra, Isha Talwar, Sayani Gupta, Mohammed Zeeshan Ayyub, Subhrajyoti Barat, Sushil Pandey, Aakash Dabhade	Sept. 6, 2019	2019	TV-MA	125 min	Dramas, International Movies, Thrillers	

localhost:8000 says

Do you really want to delete the movie 'A dummy title'?

Cancel OK

Handle form data

View definition in ***netflixapp/views.py***

```
def delete_movie(request):
    message = "No movie"
    message_type = "error"
    context = {"message": message, "message_type": message_type}
    if request.method == 'POST':
        movie_id = request.POST.get('movie-id')
        try:
            movie = Movie.objects.get(id=movie_id)
            movie.delete()
            message = f"The movie: <b>{movie.title}</b> has been deleted"
            message_type = "success"
        except ObjectDoesNotExist:
            message = f"The requested movie does not exist"
            message_type = "error"
        context = {"message": message, "message_type": message_type}
    movies = Movie.objects.order_by('title')
    context["movies"] = movies

    return render(request, 'netflixapp/tabular.html', context)
```

Handle form data

Template definition in ***netflixapp/templates/netflixapp/tabular.html***

```
{% if movies %}
   <table id="netflix-movies">
   <thead>
       <tr><th>Title</th><th>Type</th><th>Description</th>
           <th>Director</th><th>Country</th><th>Cast</th>
           <th>Date added</th><th>Release year</th><th>Rating</th>
           <th>Duration</th><th>Listed in</th><th>Delete</th></tr>
   </thead>
   <tbody>
       {% for m in movies %}
           <tr><td>{{ m.title }}</td><td>{{ m.type }}</td><td>{{ m.description }}</td>
               <td>{{ m.director }}</td><td>{{ m.country }}</td><td>{{ m.cast }}</td>
               <td>{{ m.date_added }}</td><td>{{ m.release_year }}</td><td>{{ m.rating }}</td>
               <td>{{ m.duration }}</td><td>{{ m.listed_in }}</td>
               <td><i movie-id="{{ m.id }}" movie-title="{{ m.title }}" class="lni lni-trash-can delete-button"></i>
                   <form action="delete" method="post" id="delete-form-{{ m.id }}">{% csrf_token %}<input type="hidden"
name="movie-id" value="{{ m.id }}"> </form></td>
           </tr>
       {% endfor %}
   </tbody>
   </table>
{% else %}
   <p>No movies available.</p>
{% endif %}
```

Handle form data

https://docs.djangoproject.com/en/3.2/topics/forms/#working-with-forms

UPDATE: we want to edit movies' data from our catalog.

Netflix movies

Show 10 entries

Search: Tarantino

Title	Type	Description	Director	Country	Cast	Date added	Release year	Rating	Duration	Listed in	Delete	Edit
Django Unchained	Movie	Accompanied by a German bounty hunter, a freed slave named Django travels across America to free his wife from a sadistic plantation owner.	Quentin Tarantino	United States	Jamie Foxx, Christoph Waltz, Leonardo DiCaprio, Kerry Washington, Samuel L. Jackson, Walton Goggins, Dennis Christopher, James Remar, David Steen, Dana Gourrier, Nichole Galicia, Laura Cayouette, Ato Essandoh, Sammi Rotibi, Escalante Lundy, Don Johnson	July 24, 2021	2012	R	165 min	Action & Adventure, Dramas		
dummy	Movie	Not a Tarantino movie	Myself	ITA	Myself	Nov. 25, 2021	2021	-	-	-		
Inglourious Basterds	Movie	A Jewish cinema owner in occupied Paris is forced to host a Nazi premiere, where a group of American soldiers called the Basterds plans a face-off.	Quentin Tarantino	Germany, United States	Brad Pitt, Mélanie Laurent, Christoph Waltz, Eli Roth, Michael Fassbender, Diane Kruger, Daniel Brühl, Til Schweiger, Gedeon Burkhard, Jacky Ido, B.J. Novak, Sylvester Groth, Martin Wuttke	July 22, 2019	2009	R	153 min	Action & Adventure		

Netflix movies

Edit movie

Congratulations: your movie **dummy movie** has been edited successfully!

title: dummy movie
type: Movie
description: just a dymmy movie
director: -
country: -
cast: -
date_added: dd/mm/yyyy
release_year: -
rating: -
duration: Too much
listed_in: Absolutely boring

Edit

Handle form data

View definition in ***netflixapp/views.py***

```
def edit_movie(request):
    message = "Invalid request method"
    context = {"message": message, "message_type": "error"}
    if request.method == 'POST':
        movie_id = request.POST.get("movie-id", None)
        action = request.POST.get("action", None)
        if movie_id is not None:
            # Get the movie data
            m = Movie.objects.get(id=movie_id)
            if action is not None and action == "save":
                title = request.POST.get("title", None)
                m.title = title
                # other fields here (omitted)
                m.save()
                message = f"Congratulations: your movie <b>{m.title}</b> has been edited successfully!"
                context = {"message": message, "movie": m, "message_type": "success"}
            elif action is not None and action == "edit":
                context = {"movie": m}
            else:
                message = "invalid action"
                context = {"message": message, "message_type": "error"}
        else:
            message = "Invalid movie identifier"
            context = {"message": message, "message_type": "error"}
    return render(request, 'netflixapp/edit-movie.html', context)
```

Handle form data

Template definition in ***netflixapp/templates/netflixapp/edit-movie.html***

```
<form action="edit" method="post">
<input type="hidden" name="action" value="save">
<input type="hidden" name="movie-id" value="{{ movie.id }}">
   <div class="row">
       <div class="col-1"><label>title:</label></div>
       <div class="col-4"><input type="text" name="title" id="title" value="{{ movie.title }}"></div>
   </div>
   <div class="row">
        <div class="col-1"><label>type:</label></div>
        <div class="col-4"><input type="text" name="type" id="type" value="{{ movie.type }}"></div>
   </div>
   <div class="row">
        <div class="col-1"><label>description:</label></div>
        <div class="col-4"><input type="text" name="description" id="description" value="{{ movie.description }}"></div>
   </div>
   <div class="row">
        <div class="col-1"><label>director:</label></div>
        <div class="col-4"><input type="text" name="director" id="director" value="{{ movie.director }}"></div>
   </div>

<!-- Other fields omitted due to space limitation -->
      <div class="row">
        <div class="col-5"><input type="submit" name="submit" id="submit" class="btn btn-primary" value="Edit"></div>
   </div>
</form>
```

Thank you… Happy Coding!