NAIT

SDEV 1001

Programming Fundamentals

More Loops and Exceptions - 1

A LEADING POLYTECHNIC COMMITTED TO YOUR SUCCESS

Expectations - What I expect from you

- No Late Assignments

- No Cheating

- Be a good classmate

- Don't waste your time

- Show up to class

WE ARE ESSENTIAL TO ALBERTA NAIT

Agenda

On the right is what we will cover today.

- Django URLs & Views – The Basics
- Setting Up a Django App
- Creating a Template
- Writing a View
- Adding App URLs
- Including App URLs in Project
- Passing Data to Templates
- Dynamic URL Parameters
- Handling Missing Data
- Challenge/Exercise

WE ARE ESSENTIAL TO ALBERTA | NAIT

Django URLs & Views – The Basics

- Django uses the Model-View-Template (MVT) pattern.
- A URL maps to a view, which processes a request and returns a response (often a rendered template).
- Views are Python functions or classes that handle logic for each route.

Setting Up a Django App

1. Create a new app:
   - `python manage.py startapp animal_shelter`
2. Add the app to `INSTALLED_APPS` in `settings.py`:

```
INSTALLED_APPS = [
    # ... existing apps ...
    "animal_shelter",
]
```

3. Run migrations:
   - `python manage.py migrate`

Creating a Template

- Make a directory: `animal_shelter/templates/animal_shelter/`
- Add a file: `welcome.html`
- Example content:

```html
<h1>Welcome to the Animal Shelter!</h1>
<p>Find your new best friend today.</p>
```

Writing a View

- In `animal_shelter/views.py`:

```python
from django.shortcuts import render

def welcome(request):
    return render(request, "animal_shelter/welcome.html")
```

Adding App URLs

- In `animal_shelter/urls.py`:

```python
from django.urls import path
from .views import welcome

urlpatterns = [
    path("", welcome, name="welcome"),
]
```

Including App URLs in Project

- In the project's `urls.py`:

```
from django.urls import path, include

urlpatterns = [
    path("shelter/", include("animal_shelter.urls")),
]
```

- Now, visiting `/shelter/` shows the welcome page.

Passing Data to Templates

- In `views.py`:

```
def welcome(request):
    animals = ["Dog", "Cat", "Turtle"]
    return render(request, "animal_shelter/welcome.html", {"animals": animals})
```

- In `welcome.html`:

```
<ul>
  {% for animal in animals %}
    <li>{{ animal }}</li>
  {% endfor %}
</ul>
```

Dynamic URL Parameters

- In `urls.py` :

```
path("animal/<str:name>/", animal_detail, name="animal_detail"),
```

- In `views.py` :

```
def animal_detail(request, name):
    return render(request, "animal_shelter/animal_detail.html", {"name": name})
```

- In `animal_detail.html` :

```
<h2>Details for {{ name }}</h2>
```

WE ARE ESSENTIAL TO ALBERTA

Handling Missing Data

- In `views.py`:

```
ANIMAL_INFO = {"Dog": "Friendly", "Cat": "Curious"}
def animal_detail(request, name):
    info = ANIMAL_INFO.get(name)
    return render(request, "animal_shelter/animal_detail.html", {"name": name, "info": info})
```

- In `animal_detail.html`:

```
{% if info %}
  <p>{{ name }}: {{ info }}</p>
{% else %}
  <p>No information available for {{ name }}.</p>
{% endif %}
```

WE ARE ESSENTIAL TO ALBERTA
NAIT

Challenge

- Add a page that lists animals by habitat (e.g., "land", "water").
- URL: `/habitat/<str:type>/`
- View: Filter and display animals matching the habitat type.

Summary

- Django’s URL dispatcher connects browser requests to Python view functions.
- Views handle logic, fetch or prepare data, and render templates for the user.
- Templates display data and support logic like loops and conditionals.
- Dynamic URL parameters enable flexible, data-driven pages.
- Always handle missing or unexpected data gracefully in your views and templates.
- Practice: Build pages that filter and display data based on user input or URL parameters.

Example

Let’s go run a few examples together

A LEADING POLYTECHNIC COMMITTED TO YOUR SUCCESS