SDEV 1001

Programming Fundamentals

Packages and virtual environments - 5

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

Agenda

On the right is what we will cover today.

Why Use Query Parameters?

- Query parameters let users filter, sort, and paginate data in web APIs.
- They are part of the URL, e.g., `/animals?species=dog&page=2`.
- Essential for building flexible, interactive web applications.

Setting Up: Animal Shelter API

1. **Create a virtual environment:**

```
python -m venv ./venv
.\venv\Scripts\activate
```

2. **Install Flask:**

```
pip install flask
```

3. **Run the app:**

```
flask --app shelter_app run --reload
```

Filtering Data with Query Parameters

- Example: Filter animals by species.

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/animals', methods=['GET'])
def get_animals():
    species = request.args.get('species', '')
    all_animals = load_animals('data/animals.csv')
    filtered = [a for a in all_animals if not species or a['species'].lower() == species.lower()]
    return jsonify(filtered)
```

- Try: `/animals?species=cat`

Pagination: Controlling Result Size

- Add `page` and `per_page` parameters:

```python
@app.route('/animals', methods=['GET'])
def get_animals():
    species = request.args.get('species', '')
    page = int(request.args.get('page', 1))
    per_page = int(request.args.get('per_page', 5))
    all_animals = load_animals('data/animals.csv')
    filtered = [a for a in all_animals if not species or a['species'].lower() == species.lower()]
    start = (page - 1) * per_page
    end = start + per_page
    return jsonify(filtered[start:end])
```

- Try: `/animals?species=dog&page=2&per_page=3`

Creating a New Resource with POST

- Add a new animal with a POST request:

```
@app.route('/add_animal', methods=['POST'])
def add_animal():
    data = request.get_json()
    name = data.get('name')
    species = data.get('species')
    if not name or not species:
        return jsonify({'error': 'Name and species required'}), 400
    with open('data/animals.csv', 'a', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=['name', 'species'])
        writer.writerow({'name': name, 'species': species})
    return jsonify({'message': 'Animal added!'}), 201
```

- Use Postman or curl to test POST requests.

Exercises

1. Add filtering by animal age to the `/animals` endpoint.
2. Implement sorting by name or species in the `/animals` endpoint.
3. Create a new POST endpoint to add animal adoptions to `data/adoptions.csv`.

Summary

- Query parameters allow flexible filtering and pagination in Flask APIs.
- POST requests let users add new data.
- These techniques are key for building interactive, data-driven web applications.