College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
IN608: Intermediate Application Development Concepts
Level 6, Credits 15
Practical 09 Django 3: Forms & Class-Based Views

Due Date: 31/08/2020 at 5pm

In this practical, you will complete a series of tasks covering today's lecture. This practical is worth 1% of the final mark for the IN608: Intermediate Application Development Concepts course.

Before you start, in your practicals repository, create a new branch called **09-practical**.

Task 1

Create a Django project called `dog`. `cd` to `dog`, create a virtual environment & install Django. Create an app called `practical09dogsearch`. Please ensure you configure your app in `dog/settings.py` & `dog/urls.py`. In the `practical09dogsearch` directory, create a directory called `templates` & sub-directory called `practical09dogsearch`. In `templates/practical09dogsearch`, create two HTML files called `index.html` & `results.html`.

In `models.py`, copy & paste the `Dog` model from the previous practical.

In `index.html`, create an HTML form. The form `action` should map to the `results` function in `views.py` & the `method` should be `POST`. For adaptability, friendliness, grooming needs & trainability, use radio buttons. For physical needs, use a select drop down. For each HTML form element, i.e., input & select, set the `name` attribute to the appropriate `Dog` model field & set the `value` attribute to the appropriate `RANGE_CHOICE` actual value. Add a `submit input` below the mentioned elements.

In `views.py`, create a class called `IndexView` which extends `generic.TemplateView`. In this class, set the `template_name` to `practical09dogsearch/index.html`. Create a function called `results`. In this function, you will query the `Dog` model using `filter` & return a `QuerySet` containing objects that match the given lookup arguments. In this instance, the lookup arguments will be the items in `POST` request. Render the `details.html` template with a `context` dictionary containing the `QuerySet`. In `details.html`, if the `QuerySet` in `context` is not empty, display the length of the `QuerySet` & `context` in a nicely formatted HTML table. If the `QuerySet` in `context` is empty, display an appropriate message.

In `urls.py`, set the `app_name` to `practical09dogsearch` & create two URLs which map to the `IndexView` class & `results` function in `views.py`.

Connecting to MariaDB

Thus far, you have used `SQLite` to persistently store data. Going forward, you will use the `MySQL Client` Python module & `MariaDB`. Install `MySQL Client` Python module by running the command: `pipenv install mysqlclient`. In `dog/settings.py`, change the `DATABASE` configuration to the following:
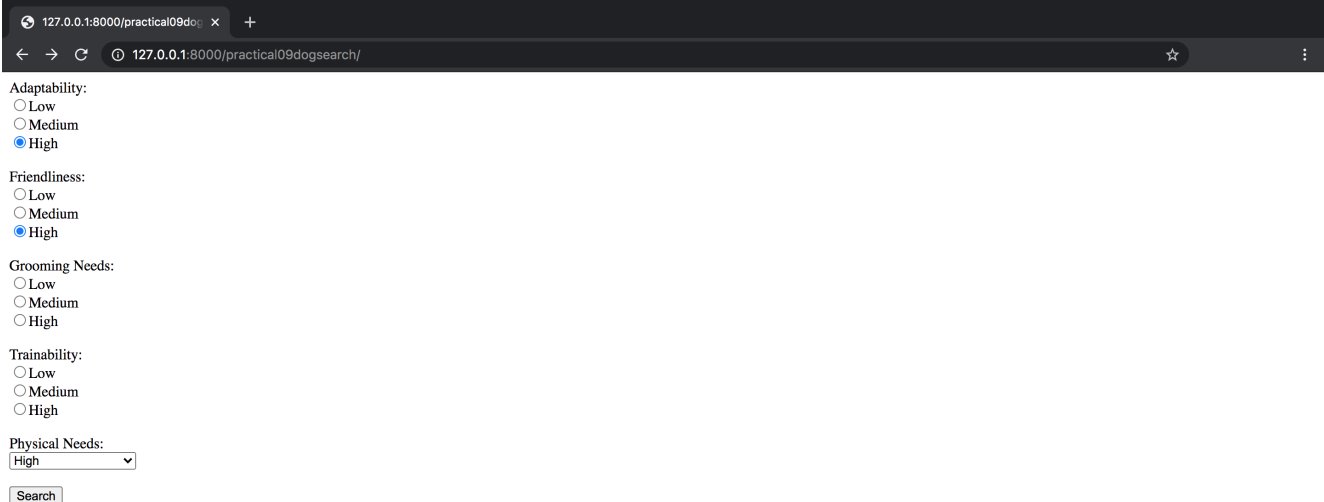
```
DATABASES = {
  'default': {
      'ENGINE': 'django.db.backends.mysql',
      'NAME': 'in608shared_<your OP username>',
      'USER': 'in608shared',
      'PASSWORD': 'P@ssw0rd',
      'HOST': 'mariadb.ict.op.ac.nz',
  }
}
```

Please ensure you run both migration commands, i.e, `python manage.py makemigrations` & `python manage.py migrate`. Go to http://mariadb.ict.op.ac.nz/phpmyadmin/ & view your `in608shared_<your OP username>` database.

Expected Output

Run the command `python manage.py runserver` then navigate to http://127.0.0.1:8000/practical09dogsearch/
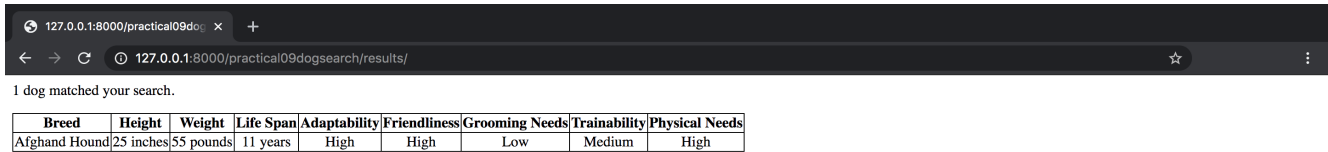
Note: The user should be able to search on any combination of fields.

Bachelor of Information Technology

1 dog matched your search.

Breed	Height	Weight	Life Span	Adaptability	Friendliness	Grooming Needs	Trainability	Physical Needs
Afghand Hound	25 inches	55 pounds	11 years	High	High	Low	Medium	High

No dogs available.

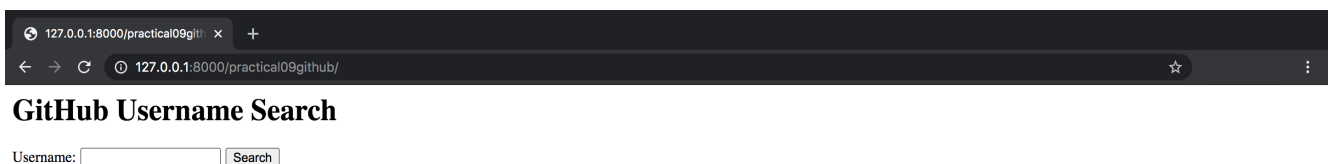**Deployment link:** https://int-app-dev-practical-09.herokuapp.com/practical09dogsearch/

Resources

- Django Queries

- MySQL Client

Task 2

Create a Django project called `github`. `cd` to `github`, create a virtual environment & install Django. Create an app called `practical09github`. Please ensure you configure your app in `github/settings.py` & `github/urls.py`. In the `practical09github` directory, create a directory called `templates` & sub-directory called `practical09github`. In `templates/practical09github`, create two HTML files called `index.html` & `details.html`.

In `index.html`, create an HTML form. The form `action` should map to the `details` function in `views.py` & the `method` should be `POST`. Add a `text input` with the `name` attribute value of `username` & `submit input`.

GitHub Username Search

Username: [   ] [Search]

In `views.py`, create a class called `IndexView` which extends `generic.TemplateView`. In this class, set the `template_name` to `practical09github/index.html`.
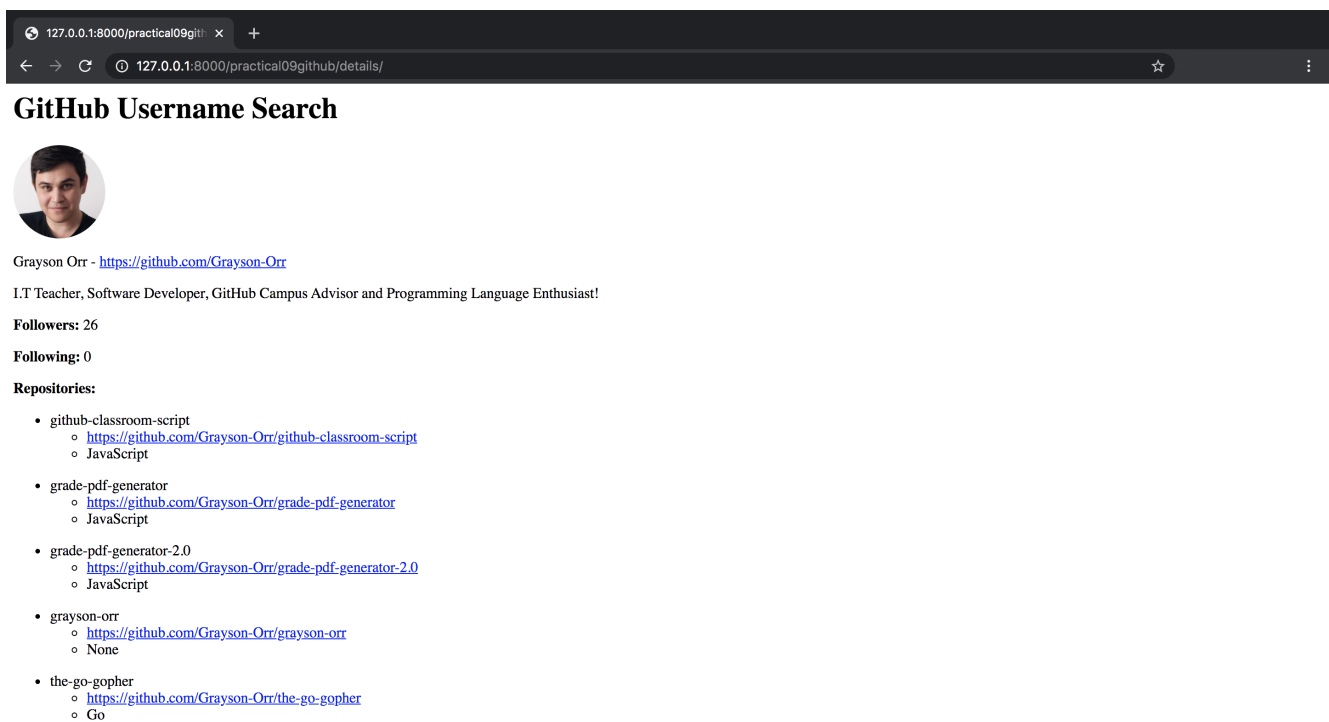
Create a function called `details`. In this function, get the `POST POST username` from the form. Use the `POST username` value & make a `GET` request to `https://api.github.com/users/<username>`. **Note:** replace `<username>` with the `POST username` value. Please view the example response: https://api.github.com/users/grayson-orr
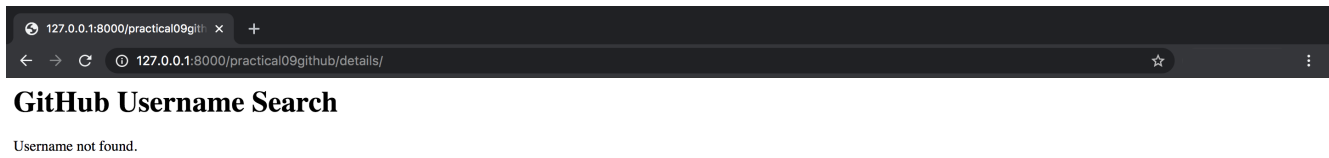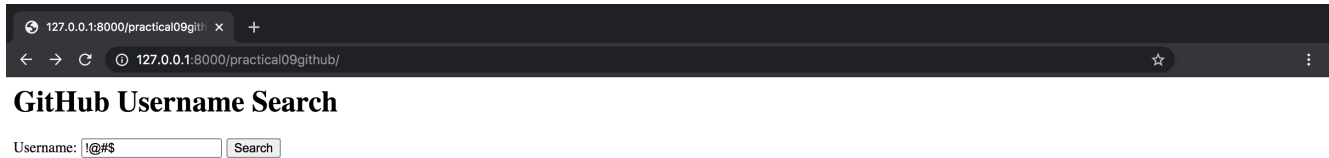
Version 1, 2020

3

Create a dictionary called `context`. If a `POST username` value is found, add the response contents from the `GET` request to the dictionary. In addition, make another make a `GET` request to `repos_url`. The response content will contain a list of repository information for that particular **GitHub** user. Add the response content to `context`. Please ensure correct error checking.

In `details.html`, display the `context` dictionary in the same format as shown in the expected output.

Expected Output

Run the command `python manage.py runserver` then navigate to http://127.0.0.1:8000/practical09github/

GitHub Username Search

Username: !@#$ Search

GitHub Username Search

Username not found.

Deployment link: https://int-app-dev-practical-09.herokuapp.com/practical09github/

Resources

- GitHub Developer API