CINS 465: Project 2: Organizer w/REST API & Containerization

This is an individual assignment. **Do your own work. Do not copy / paste anything from any outside source** You may brainstorm with or get help on technical problems from others, but all submitted work must be your own. It is highly recommended that you use Piazza as a tool for remote group collaboration. In addition to asking questions and reading answers on Piazza, please read other's questions and post answers if you know them.

Objectives

- Use the Django REST framework to create a hyperlinked REST API for Organizer.
- Create a Docker container for Organizer.

Requirements

- Expose the following set of hyperlinked REST end-points for accessing the related organizer entities:
  - /api/v1/users
  - /api/v1/tasks
  - /api/v1/task-categories
  - /api/v1/budget
  - /api/v1/budget-categories
  - /api/v1/journal
- Add a navigation item named REST-API that links to your main REST endpoint, /api/v1/

General Requirements

- You must expose a hyperlinked REST API like the one I demo'd in lecture on Tues. April 7th, with all of the REST endpoints mentioned above.
- You must use the Django REST framework (this makes the work much easier).
- **You must submit a zip or tar file** containing your entire Django project such that when I open it I can navigate into your project directory and start your web server using manage.py. Your Django project can be named whatever you want (it doesn't need to be named project2).
- Note: It doesn't matter what the name of your actual Django project is within your docker container. Because Django projects are difficult to rename, and because you will be basing this submission off of what you did for Project 1, it is perfectly fine for the name of your django project to be **project1**, or anything else, like maybe **organizer** or whatever you called it originally.
- **You must submit a compressed tar file** named project2.docker.tgz, containing a docker image named **project2**, containing your django project.
- Your Docker image must work properly when I run these three docker commands, and then load the URL http://127.0.0.1:8000/
  - **docker load < project2.docker.tgz**
  - **docker run -it -p 8000:8000 project2**

Suggested "Plan of Attack"

- Setup / Prep
  - Checkpoint your Project 1 source code (push the latest to github).
  - Download my example Project 1 diffs that shows the changes needed to implement a REST Framework for Users, Tasks and Task-Categories: Within our Blackboard section: Course Content > Lectures > Week 11 > Example DRF Diffs for Project 1
  - Note: You don't need to create a new Django project, or rename your existing one, just keep using the name you used for Project 1.
  - Create and activate a new conda environment (just in case something goes wrong with the Django REST framework install).
    - **conda create --name env_drf**
    - **conda activate env_drf**
  - Install Django and the Django REST framework.
    - **conda install django**
    - **conda install -c conda-forge djangorestframework**
  - Skim through the background info on DRF, here: https://www.django-rest-framework.org/#
- Implement your REST APIs
  - Add 'rest_framework into INSTALLED_APPS, in settings.py
  - Create DRF serializers

        - Create file searializers.py within application (tasks application in my example)
        - Create new serializers for Task, TaskCategory, User etc.
    - Create rest_framework ViewSets in the various views.py files for your applications.
        - See posted example diffs mentioned above for how I did this.
        - See https://www.django-rest-framework.org/api-guide/viewsets/ for background info on DRF ViewSets.
    - Configure rest_framework router, and REST api paths in urls.py
        - See posted example diffs mentioned above for how I did this.
        - See https://www.django-rest-framework.org/api-guide/routers/ for background info on DRF Routers.
    - Add Navigation Item for REST API end-point (/api/v1/).
    - Test all of the above and call this checkpoint 1. Now would be a good time to commit your changes and push to github.
- Containerize your Django project.
    - Skim the Docker homepage, here: https://www.docker.com/
    - Install Docker on your local development machine
        - https://docs.docker.com/install/
        - For Windows 10 WSL2 users, see also https://docs.docker.com/docker-for-windows/wsl-tech-preview/
    - Create your project2 Docker image
        - From inside your main django project directory:
            - **conda env export --no-builds > environment.yml**
            - Rename conda environment listed at top of environment.yml to 'env'
            - Create a new file named **run_server.sh**, with the following contents:

```
#!/bin/sh
python app/manage.py runserver 0.0.0.0:8000
```

            - Create a new file named **Dockerfile**, with the following contents:

```
FROM continuumio/miniconda:latest
COPY environment.yml ./
COPY . app
COPY run_server.sh ./
RUN chmod +x run_server.sh
RUN conda env create -f environment.yml
RUN echo "conda activate env" > ~/.bashrc
ENV PATH /opt/conda/envs/env/bin:$PATH
EXPOSE 8000
ENTRYPOINT ["./run_server.sh"]
```

            - Build a docker image named **project2** from **Dockerfile**
                - **docker build -t project2 .**
            - Create, run & test a docker container from your docker image
                - **docker run -it -p 8000:8000 project2**
                - You should see output similar to when you start your Django projects manually, that is, something similar to:

```
Django version 3.0.3, using settings 'project1.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

                - Note: It is normal / expected to see http://0.0.0.0:8000/ instead of the usual http://127.0.0.1:8000/
            - Load URL http://127.0.0.1:8000/ from your local browser and make sure your site is working as expected.
            - Call this checkpoint 2. Now would be a good time to commit and push to github.
        - Create a compressed tar file containing your project 2 docker image
            - **docker save project2 | gzip > project2.docker.tgz**
        - Create a tar or zip file containing just your Django project, the same way you did for Project 1.

- No need to create a demo video for Project 2, you'll get to do that again for the final project.