Heroes Villains API

Learning Objective

Build a REST web API in Django REST Framework utilizing multiple apps and foreign key model relationships.

Technologies

Python, Django REST Framework, Postman

User Stories

Total Points: /47.5

Total Wtd Points: /20

(/5 points): As a developer, I want to make good, consistent commits.

(/2.5 points) As a developer, I want to create a SuperType model in a “super_types” app.
Property names must be in snake_case and match the following exactly!

- type – CharField

(/5 points) As a developer, I want to register the SuperType model with the admin site so I can:

1. Register a new super user (python manage.py createsuperuser)
2. Visit the admin site
3. Seed two values (“Hero” and “Villain”) into the “super_type” table

(/2.5 points) As a developer, I want to create a Super model in a “supers” app.
Property names must be in snake_case and match the following exactly!

- name - CharField
- alter_ego - CharField
- primary_ability - CharField
- secondary_ability – CharField

- catchphrase – CharField
- super_type – ForeignKey

(/2.5 points) As a developer, I want my API to serve the "**supers**" app's content on the following urls paths:
Paths must match these exactly!

- **'127.0.0.1:8000/api/supers/'** - optional params
- **'127.0.0.1:8000/api/supers/<int:pk>/'**

(/5 points) As a developer, I want to create a GET by id endpoint that does the following things:

- Accepts a value from the request's URL (The id of the super to retrieve).
- Returns a 200 status code.
- Responds with the super in the database that has the id that was sent through the URL.

(/5 points) As a developer, I want to create a POST endpoint that does the following things:
Accepts a body object from the request in the form of a Super model.

- Adds the new super to the database.
- Returns a 201 status code.
- Responds with the newly created super object.

(/5 points) As a developer, I want to create a PUT endpoint that does the following things:

Accepts a value from the request's URL (The id of the super to be updated).

Accepts a body object from the request in the form of a Super model.

- Finds the super in the Super table and updates that super with the properties that were sent in the request's body.
- Returns a 200 status code.
- Responds with the newly updated super object.

(/5 points) As a developer, I want to create a DELETE endpoint that does the following things:

- Accepts a value from the request's URL.
- Deletes the correct super from the database
- Returns a 204 status code (NO CONTENT).

(/10 points) As a developer, I want to create a GET endpoint the responds with a 200 success status code and all of the supers within the supers table.

- This view function should be implemented in a way to accept a “type” parameter
    - Example: " http://127.0.0.1:8000/api/supers?type=hero”
- **If a type query parameter is sent to the view function with the value of “hero”, the view function response should be a list of all supers that are associated with the type of “Hero” (Shown in End Result Overview video on portal)**
- **If a type query parameter is sent to the view function with the value of “villain”, the view function response should be a list of all supers that are associated with the type of “Villain” (Shown in End Result Overview video on portal)**
- **If no type query parameter is sent, return a custom dictionary response with a “heroes” key set equal to a list of supers of type “Hero” and a “villains” key set equal to a list of supers of type “Villain” (Shown in End Result Overview video on portal)**
    - **Custom_response = {“heroes” = [], “villains” = []}**