Django Treasure Hunt

@CodeRun @Mali Imre Gergely

Technical specifications:

- Python
- Django
- HTML / CSS / JS
- any relational DB

Description

Modern problems require modern solutions, and modern pirates use Django apps to find treasure and gold. This time you are a pirate (not the one using torrents and stuff but like an actual one) seeking for treasure and you will write a Django app to help you on this journey.

Easy version:

This version will consist of a page at path **'/compass'.** This will show a form that will allow the pirate to add instructions to the database. Each instruction has:

- ID - a unique number
- Direction - One of east, west, south, north, north-east, north-west, south-east, south-west. This one should be a dropdown
- Distance in nautical miles (int, greater than zero)
- Description - nonempty string
- Previous Instruction - each instruction will be linked to a previous one except for the first so that a route (or multiple ones for that matter) can be formed. Read about models for some inspiration.
- Risk Level - a number between 0 and 100 marking how risky the seas will be after taking this turn

Below the form, an HTML list will show all the existing instructions. Each row will contain the direction and distance of the instruction. (Example, 14 miles north-east)

A *Details* button will appear next to each actor as part of its row, clicking on which the user will be redirected to the **'instructions/<instructionID>'** path, which will open a page with the corresponding instruction, showing all the details of the instruction, including rendering the direction as a picture/svg.(Hint: use svgs like this one for north, this one for north-east etc. or simple arrows or any other visual as long as it is visible, interpretable and consistent).

Furthermore, on each instruction detail page, a Summary button will appear. When the user presses that button, the page will display a summary with the following details: considering that the treasure hunt starts from the instruction of the current page, the average risk level of the route until the end, the total distance.

- Data can be saved in any relational DB
- Views, urls, models, and templates will be configured in their proper places
- The form will be validated for errors

Hard part

This part is more interactive as it will walk you through finding the treasure. We start from the shore only having a compass and an insatiable hunger for gold. This description you are reading will help you start off and set sail but further instructions you will find by using your compass and your oracle at this host. Let's start by using the compass.

The **'/compass'** page will contain a form with one input:
The instruction ID. Upon submission, you will receive the next direction on your route, along with descriptions and explanations at the following path of the provided host: 'direction/**set-sail**' (revise urls to see how to do that). Note that this has to be combined with the provided oracle host and not part of your app. You can change the bold part according to the instructions you will be given. You will have to download and save each direction into a db and display the Title, cardinal direction and distance. Templates will be your friend on this quest. Fill in the ID of the last instruction to get the next one and submit the form. Also render the picture you get for each phase of your journey. In the end, the oracle will tell you how to proceed. Just read those instructions.
Hints: use requests to connect to the oracle API

Points:
Easy:

- Attendance: 10 points
- Code style, readability: 10 points
- Add functionality: 10 points
- Form validation: 5 points
- DB config: 10 points
- Url configuration: 10 + 15 points
- Displaying the instructions + template on the main page: 20 points
- Details page 20 points

Hard:

- Attendance: 20 points
- Code style, readability: 15 points
- Main page: 25 points
- DB config with proper relations: 25 points
- Compass page, API call: 25 points
- Finding the treasure, instructions, and chest code: 110 points