MIT AITI
Python Software Development

Lab 3: Setting up a Django Website for Web Apps

In this lab, you will get hands-on experience in using Django to create your own website. This will be helpful for designing web apps that will access data from your website. Follow the steps below to get started.

Setup:
You will need the following software installed.
1. Eclipse
2. Android SDK
3. ADT plug-in (to use the Android SDK in Eclipse)
4. EGit plug-in (to use Git through Eclipse)

Steps:
1. You will be working with the TravelIndia project that we worked with in class. You will need a new copy of the project, since you had made some changes to the project during class. This means you will need to delete your current TravelIndia project and then import it again.
    a. In Eclipse, right-click on your current TravelIndia project. Select “Delete”.
    b. Make sure you check the box next to “Delete project contents on disk (this cannot be undone)”. Then press “OK”.
    c. Go to File -> Import -> General -> Existing projects into workspace. Then click “Next”.
    d. Choose “Select archive file”. Click “Browse” and navigate to wherever you had saved the TravelIndia.zip file. Click “Finish”
    e. You should see a new TravelIndia project in the workspace.
2. Create a new model called Attraction in the Places application.
    a. Navigate to models.py in the Places folder.
    b. Create a new class called Attraction.
    c. Add the properties of name, city, latitude, and longitude to Attraction. Name and city should be stored as CharFields. Latitude and longitude should be stored as FloatFields.
    d. Add the property of state. This is the state that the Attraction belongs to. Make sure the state is stored as a Places.State object, not as a string.
    e. Call sync DB (Right-click on project -> Django -> sync DB), so that Attractions are added to the database.
3. List all the attractions in a state on the state details page (details.html)
    a. First you will need to modify the show_state view. Go to the views.py file in the Places folder.
    b. Currently, the method “show_state” passes a State object to the Context. You will add to this method so that it also passes a list of the attractions in the state to the Context. Call the list “attractions”. Make sure you pass a list of Attraction objects, not strings.
    c. Now you can modify the actual state details page to display the attractions. Find the file details.html. It is located in templates -> places -> states -> details.html. Right-click on this file, and select Open With -> Text Editor.

    d. Add code to the html file to make it display the heading “Attractions” below “Population Density”.
    e. Add code to display the list of attractions under the heading “Attractions”. Remember you can access the list of Attraction objects through the variable “attractions” that you passed in through the show_state view.
    f. To test the changes you’ve made to the State Details page, you will have to make some attractions and save them to the database. Open the shell with Django environment (right-click on project -> Django -> shell with Django environment). Then create a couple Attraction objects and save them. Make at least 5 attractions in Maharashtra.
    g. Now run the server (right-click on project -> Django -> custom command -> type “runserver”). Open your browser and type “localhost:8000/states”. You should see a list of Indian states. If you click on one of the states, you will go to the State Details page. Here you should see the heading Attractions, as well as the names of any Attractions that you made.
4. Link to a page that shows Attraction details for each of your Attractions.
    a. Navigate to views.py in the Places folder. Create a view similar to show_state called show_attraction. This method should take in state_id and attraction_id as parameters. Make sure you pass the Attraction object to the Context. The name of the template should be ‘places/states/attraction_detail.html’.
    b. Now you must create the page that will display attraction details. Navigate to templates -> places -> states. In this folder, create a new html file called attraction_detail.html.
    c. Add code to attraction_detail.html, which will allow for all of the attraction details to be displayed. Remember that the attraction details are name, city, latitude, and longitude. See how state details are displayed in detail.html for guidance.
    d. Go back to detail.html. Convert the attraction names into links that link to the attraction details page. The links should be of the form “attractions/<attraction_id>”.
    e. Finally, navigate to the file called urls.py. Add a line similar to the others that will allow a url of the form /states/<state_id>/attractions/<attraction_id> to invoke the show_attraction view.
    f. Now if you reload the server, and refresh your web page, you should be able to click on any of your attractions and go to a page that shows details about that attraction.
5. Finally, you will additionally add views that return the list of attractions in a state and the attraction details in JSON.
    a. Modify the view called “show_state_JSON” in views.py, so that it adds the attraction data. This is similar to how you modified “show_state” in step 3b. You can just pass a list of attraction names, not the Attraction objects.
    b. Create a view called “show_attraction_JSON” in views.py. This method should pass all the attraction detail data to JSON. Look at show_state_JSON for guidance.
6. You should have a working website that allows users to learn more about the states and attractions in India! Now you could design a web app that would access the data from this website.