

django_assignment_001.md

Building Web App with Django


Let's build a Question App.

Build the following views to make your web app functional:

- `get_list_of_questions`
- `create_question`
- `get_question`
- `update_question`
- `delete_question`

Write all your code in the `questions` app given to you.

You need to write:

- `urls`: Each view has a URL specified. Follow the exact URL naming.
- `views`: Write all your views in `views.py`  file of the `questions` app. Give your views the exact name mentioned at each view discussed below
- `templates`: Templates for each of the below views are already present in the app given to you. You need to make slight changes with the knowledge you have learned in the Django tutorials to make them fully functional.
- `models`: Write `Question` model as mentioned below with the exact field and model names
 - model name: `Question`
 - fields:
 - `text` - `TextField`
 - `answer`- `TextField`

Suggested Process:

- Write all the URLs first.
- Start with writing dummy views by rendering the suggested templates at each view (if there are more than one then choose any one of them)

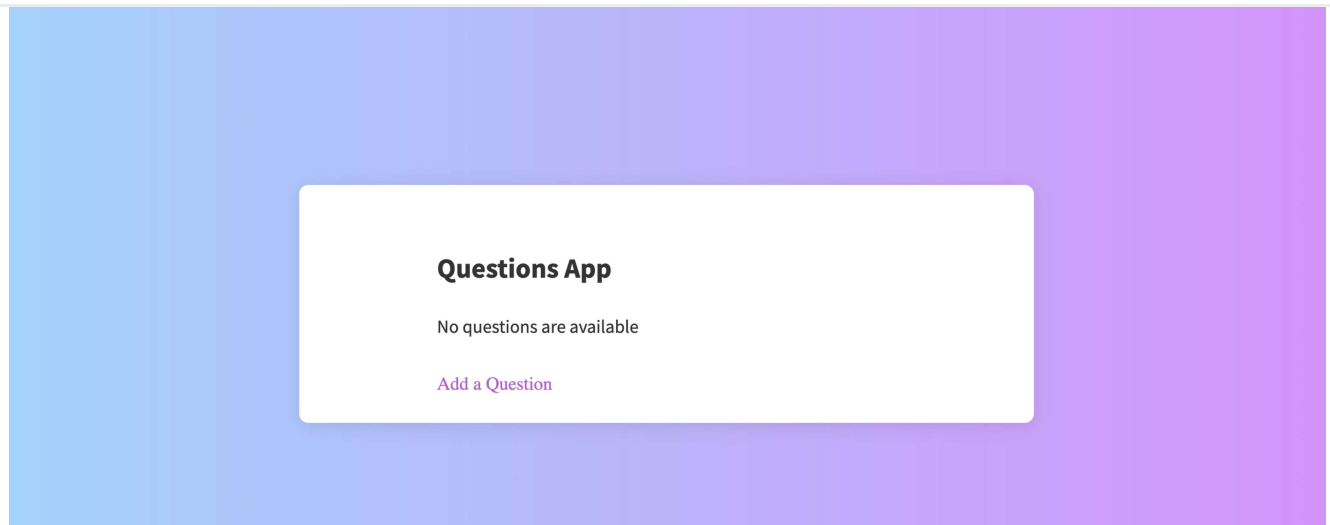
≡ iBHubs

images provided at each view

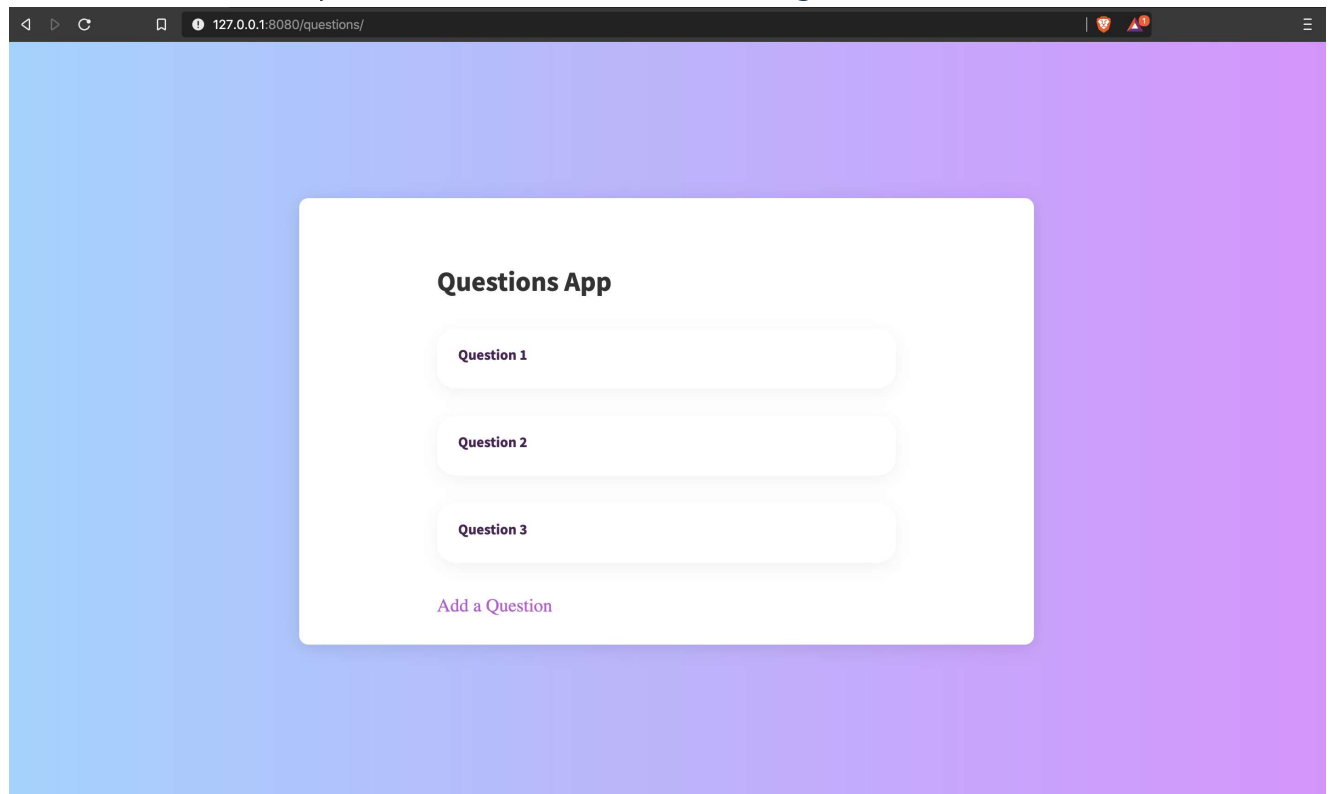
- Separate the differentiation into two categories
 - logic in the view
 - logic in the template
- Logic in View:
 - Filling templates with data from the database
 - First list down all the data needed
 - Pass appropriate data to the `render` method
 - Render response based on the data in the request
 - Use appropriate methods/attributes to access data in the request body
 - **Get HTTP Method** [↗](#)
 - **Accessing data in body** [↗](#)
- Logic in Template:
 - Depending on the scenario use one of the following
 - Using if-else block
 - Using loops
 - When making HTTP Request to a view from templates, double-check the URLs. Properly pass `question_id` in URL paths where required.

Get list of Questions

- View name: `get_list_of_questions`
- URL: **<http://localhost:8080/questions/>** [↗](#)
- When there are no questions in the database, your view should return the following HTTP Response



- When there are questions in the database, your view should return all the questions in the database as HTTP Response (as shown in the below image)

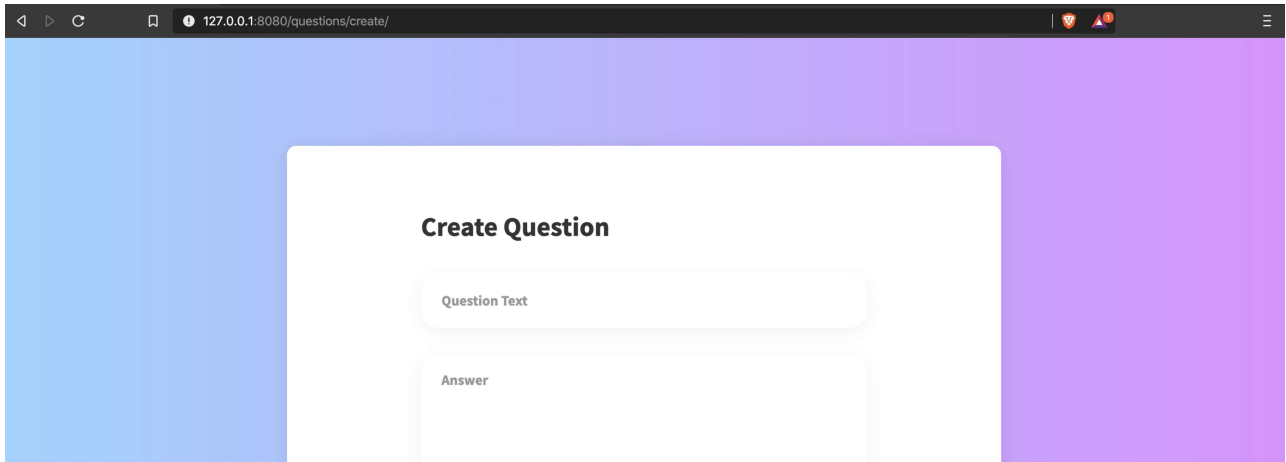


- Make use of `get_list_of_questions.html` template given in the `questions` app.

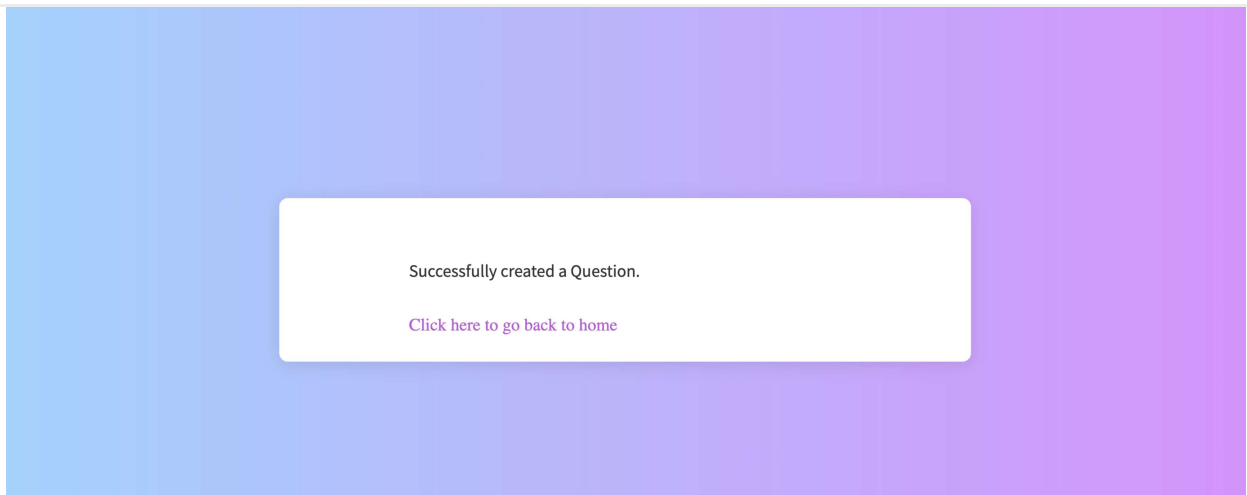
Create Question

≡ iBHubs

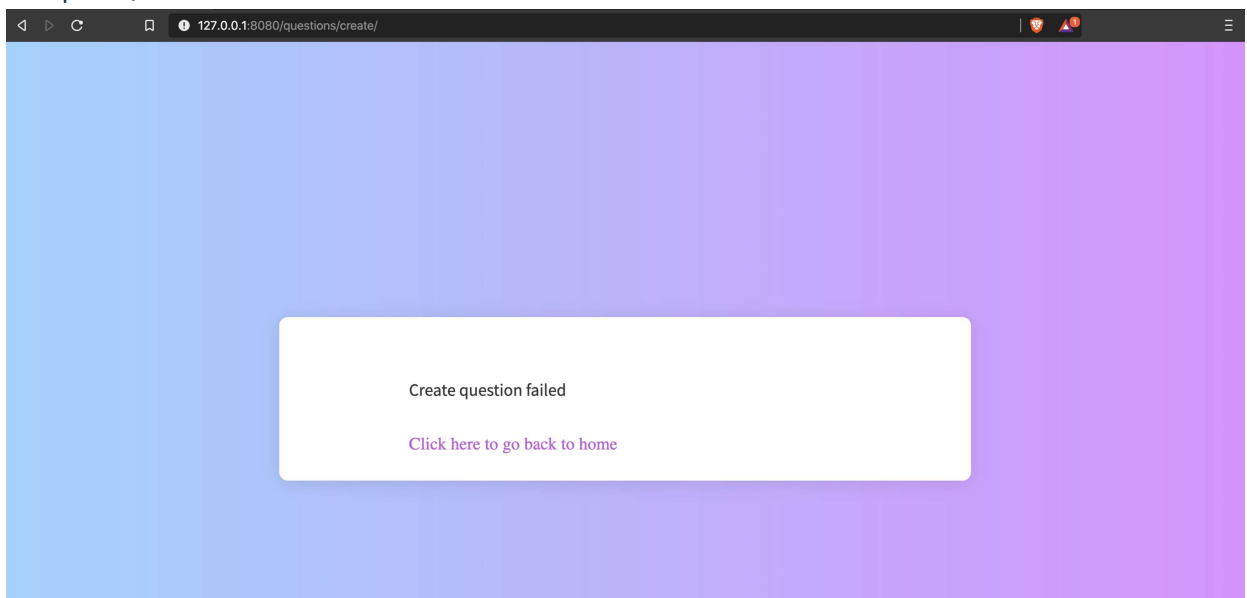
- This question has two parts
- Part1
 - When an HTTP Request with the GET method is sent on the above url, Your view is expected to show the below form so that users can create new questions (use `create_question_form.html` template)

A screenshot of a web browser window. The address bar shows the URL '127.0.0.1:8080/questions/create/'. The page has a light blue and purple gradient background. In the center, there is a white card titled 'Create Question'. Below the title, there are two text input fields. The first field is labeled 'Question Text' and the second field is labeled 'Answer'. Both fields are empty.

- `create_question_form.html` is **already coded (you need not make any changes in this template)** in a way that on clicking save button an HTTP Request with the POST method is sent on the above URL along with the data in the given form.
- Part 2
 - When an HTTP request with the POST method is sent to the above URL. Your view is expected to save the data (body of the HTTP Request) in the database if it's valid (i.e. question and answer should be nonempty strings).
 - In response
 - If given data is valid, send a success response (use `create_question_success.html` template)

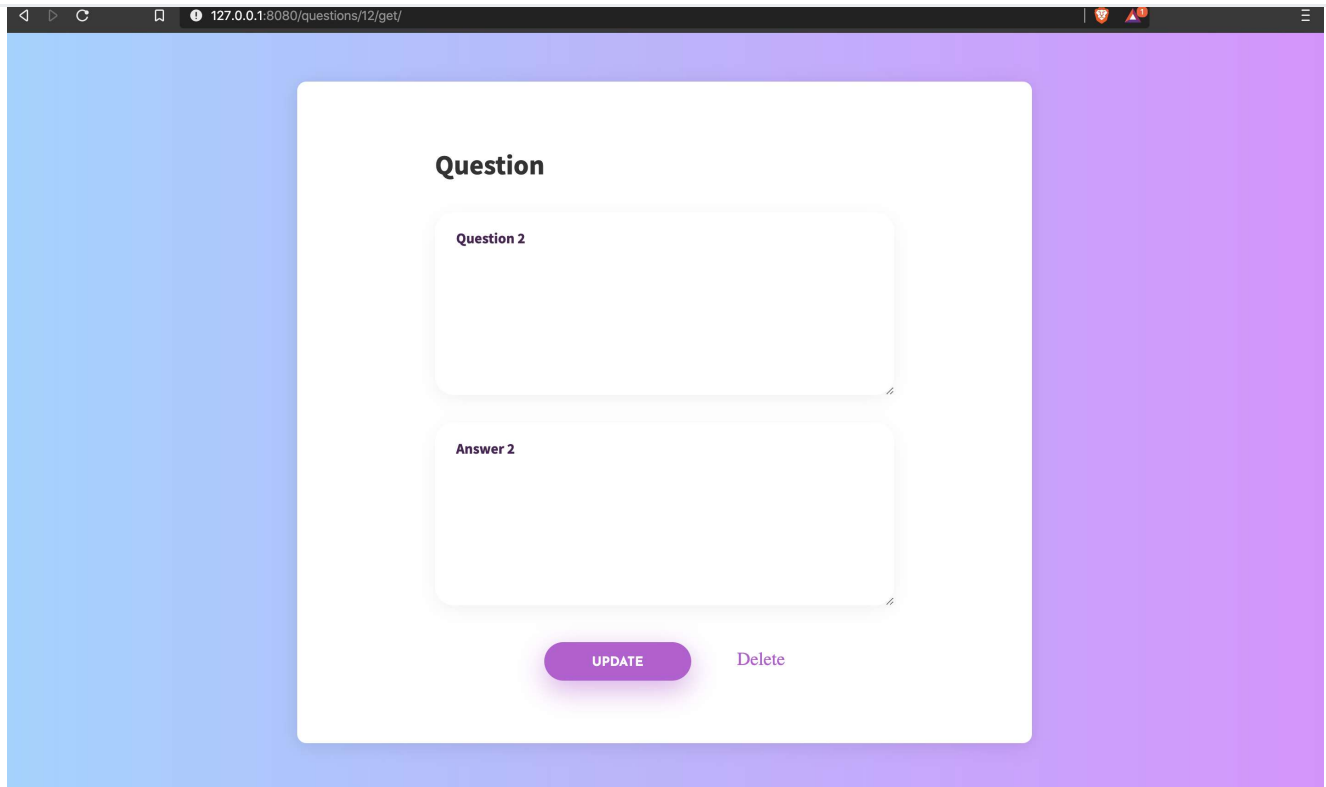


- If given data is invalid, send a failure response (use `create_question_failure.html` template)




Get Question

- View name: `get_question`
- URL: http://localhost:8080/questions/{question_id}/get/



- This view will be called when you click a question on the home page.

Update Question

- View name: `update_question`
- URL: http://localhost:8080/questions/{question_id}/update/ 
- On clicking the update button in the `Get Question` view, an HTTP request is made to the above URL
- Your view is expected to save this data in the database if it's valid (i.e., question and answer should be nonempty strings).
- In response, users should be shown a success response in case of success else a failure response.

≡ iBHubs

- Failure Response (use `update_question_failure.html` template)



Delete Question

- View name: `delete_question`
- URL: http://localhost:8080/questions/{question_id}/delete/ 

≡ iBHubs

valid(i.e., a question with given `question_id` is in the database).

- In response, users should be shown a success response in case of success else a failure response.
 - Success Response (use `delete_question_success.html` template)
 - Failure Response (use `delete_question_failure.html` template)

Additional Questions [Optional]

Sort Questions

- View name: `get_list_of_questions`
- URL: <http://localhost:8080/questions/>
- Your view should accept `sort_by` query parameter (or query string) if the value for the query parameter(query string) is
- `asc` your view should show the questions in ascending order of question text.
- `desc` your view should show the questions in descending order of question text. If the values
- If no value is passed to the query parameter, then sort then in ascending order of question text
- You can make use of the below html code for this question

```
<form action="/questions">
    <input type="hidden" name="sort_by" value="asc">
    <input type="submit" value="Sort by ASC" class="contact100-form-btn"
        formmethod="get">
</form>
```

html