|  |
| --- |
|  |
|  | **1) CreateURL Mapping for the respective project defined** |
|  | **II. Keep admin URL separately** |
|  | **III. Use templates for the URL connections** |
|  | **IV. Define static folder and add CSS files.** |
|  | **V. Use Bootstrap modules** |
|  | **VI. Change the admin interface as per your project statements.** |
|  | **2) Create a ModelForm for AUTHOR{name, book and bookid}** |
|  | **Add the table &lt;appname&gt;\_author into the database.** |
|  | **3) Render HTML Forms (GET &amp; POST) in Django for the fields fist name and last name** |
|  | **4) Use Django built-in methods to validate form login page data automatically. Use CSRF tokens.** |
|  |  |
|  | **Answers :** |
|  |  |
|  | **I. URL Mapping** |
|  | 1. Home Page: / |
|  | 2. Authors: /authors/ |
|  | 3. Author Details: /authors/<author\_id> |
|  | 4. Admin Login Page: /admin/login |
|  | 5. Admin Page: /admin/ |
|  |  |
|  | II. Admin URL Mapping |
|  | 1. Admin Home Page: /admin/ |
|  | 2. Admin Author List: /admin/author/list |
|  | 3. Admin Author Create: /admin/author/create |
|  | 4. Admin Author Update: /admin/author/update/<author\_id> |
|  | 5. Admin Author Delete: /admin/author/delete/<author\_id> |
|  |  |
|  | III. Templates |
|  | 1. Home Page: templates/index.html |
|  | 2. Author List: templates/authors.html |
|  | 3. Author Details: templates/author\_detail.html |
|  | 4. Admin Login Page: templates/admin/login.html |
|  | 5. Admin Home Page: templates/admin/index.html |
|  | 6. Admin Author List: templates/admin/author\_list.html |
|  | 7. Admin Author Create: templates/admin/author\_create.html |
|  | 8. Admin Author Update: templates/admin/author\_update.html |
|  | 9. Admin Author Delete: templates/admin/author\_delete.html |
|  |  |
|  | IV. Static Folder |
|  | Add the below files to the static folder |
|  | 1. CSS File: static/css/main.css |
|  | 2. Bootstrap File: static/bootstrap/bootstrap.min.css |
|  | 3. JS File: static/js/main.js |
|  |  |
|  | V. Use Bootstrap Modules |
|  | 1. Use the Bootstrap Grid System, Forms, Buttons, and Navbars. |
|  | 2. Use the Bootstrap components like Alerts, Modals, and Dropdowns. |
|  |  |
|  | VI. Change Admin Interface |
|  | 1. Add an Admin Panel page to manage Authors. |
|  | 2. Create a separate page to add, update and delete Authors. |
|  | 3. Add additional functionality such as sorting, filtering and pagination. |
|  |  |
|  |  |
|  |  |
|  | **2) Create a ModelForm for AUTHOR{name, book and bookid}** |
|  | **Add the table <appname>\_author into the database.** |
|  |  |
|  | class AuthorForm(ModelForm): |
|  | class Meta: |
|  | model = Author |
|  | fields = ['name', 'book', 'bookid'] |
|  |  |
|  |  |
|  |  |
|  |  |
|  | 3) Render HTML Forms (GET & POST) in Django for the fields fist name and last name |
|  |  |
|  | &lt;form action="" method="post"&gt; |
|  | {% csrf\_token %} |
|  | &lt;div class="form-group"&gt; |
|  | &lt;label for="firstName"&gt;First Name:&lt;/label&gt; |
|  | &lt;input type="text" class="form-control" name="firstName" id="firstName"&gt; |
|  | &lt;/div&gt; |
|  | &lt;div class="form-group"&gt; |
|  | &lt;label for="lastName"&gt;Last Name:&lt;/label&gt; |
|  | &lt;input type="text" class="form-control" name="lastName" id="lastName"&gt; |
|  | &lt;/div&gt; |
|  | &lt;button type="submit" class="btn btn-primary"&gt;Submit&lt;/button&gt; |
|  | &lt;/form&gt; |
|  |  |
|  |  |
|  |  |
|  | 4) Use Django built-in methods to validate form login page data automatically. Use CSRF tokens. |
|  |  |
|  | &lt;form action="" method="post"&gt; |
|  | {% csrf\_token %} |
|  | &lt;div class="form-group"&gt; |
|  | &lt;label for="firstName"&gt;First Name:&lt;/label&gt; |
|  | &lt;input type="text" class="form-control" name="firstName" id="firstName"&gt; |
|  | {% if form.errors.firstName %} |
|  | &lt;div class="alert alert-danger"&gt; |
|  | {{ form.errors.firstName }} |
|  | &lt;/div&gt; |
|  | {% endif %} |
|  | &lt;/div&gt; |
|  | &lt;div class="form-group"&gt; |
|  | &lt;label for="lastName"&gt;Last Name:&lt;/label&gt; |
|  | &lt;input type="text" class="form-control" name="lastName" id="lastName"&gt; |
|  | {% if form.errors.lastName %} |
|  | &lt;div class="alert alert-danger"&gt; |
|  | {{ form.errors.lastName }} |
|  | &lt;/div&gt; |
|  | {% endif %} |
|  | &lt;/div&gt; |
|  | &lt;button type="submit" class="btn btn-primary"&gt;Submit&lt;/button&gt; |
|  | &lt;/form&gt; |
|  |  |
|  |  |
|  |  |
|  | Model form |
|  |  |
|  | 1) Create a model form and write the CRUD queries in the dijango shell |
|  | class sampleModel(models.Model): |
|  | # fields of the model |
|  | title = models.CharField(max\_length = 200) |
|  | description = models.TextField() |
|  | last\_modified = models.DateTimeField(auto\_now\_add = True) |
|  |  |
|  | 2) Write a program to upload the txt file and check in the admin view. |
|  | 3) Create an Employee model(Ename, EID) and write the into csv file using |
|  | Dynamic CSV and database |
|  | 4) Get a string message &quot;Hello How are you &quot; from the user from and print into the pdf |
|  | file. |
|  |  |
|  | Answers |
|  |  |
|  | 1) |
|  |  |
|  | # Create a form for the model |
|  |  |
|  | from django import forms |
|  |  |
|  | class SampleModelForm(forms.ModelForm): |
|  | class Meta: |
|  | model = SampleModel |
|  | fields = '\_\_all\_\_' |
|  |  |
|  | # CRUD Queries |
|  |  |
|  | # Create |
|  |  |
|  | # Create a new object |
|  | new\_object = SampleModel.objects.create(title='My object', description='My description') |
|  |  |
|  | # Read |
|  |  |
|  | # Get all objects |
|  | all\_objects = SampleModel.objects.all() |
|  |  |
|  | # Get an object by id |
|  | object\_by\_id = SampleModel.objects.get(id=1) |
|  |  |
|  | # Get objects by filter |
|  | filtered\_objects = SampleModel.objects.filter(title='My object') |
|  |  |
|  | # Update |
|  |  |
|  | # Update an object |
|  | updated\_object = SampleModel.objects.get(id=1) |
|  | updated\_object.title = 'New title' |
|  | updated\_object.save() |
|  |  |
|  | # Delete |
|  |  |
|  | # Delete an object |
|  | deleted\_object = SampleModel.objects.get(id=1) |
|  | deleted\_object.delete() |
|  |  |
|  | 2) |
|  | from django.shortcuts import render |
|  | from django.http import HttpResponse |
|  | from django.views.decorators.csrf import csrf\_exempt |
|  | from django.conf import settings |
|  |  |
|  | import os |
|  |  |
|  | @csrf\_exempt |
|  | def upload\_file(request): |
|  | if request.method == 'POST': |
|  | file = request.FILES['file'] |
|  | filename = file.name |
|  | path = os.path.join(settings.MEDIA\_ROOT, filename) |
|  | file\_url = os.path.join(settings.MEDIA\_URL, filename) |
|  | with open(path, 'wb+') as f: |
|  | for chunk in file.chunks(): |
|  | f.write(chunk) |
|  | return HttpResponse(file\_url) |
|  |  |
|  | # urls.py |
|  | from django.urls import path |
|  | from . import views |
|  |  |
|  | urlpatterns = [ |
|  | path('admin/upload/', views.upload\_file, name='upload\_file'), |
|  | ] |
|  |  |
|  | 3) |
|  |  |
|  |  |
|  | #models.py |
|  | from django.db import models |
|  |  |
|  | class Employee(models.Model): |
|  | ename = models.CharField(max\_length=100) |
|  | eid = models.IntegerField() |
|  |  |
|  | #views.py |
|  | import csv |
|  | from django.http import HttpResponse |
|  |  |
|  | def export\_employee\_csv(request): |
|  | response = HttpResponse(content\_type='text/csv') |
|  | response['Content-Disposition'] = 'attachment; filename="Employee.csv"' |
|  |  |
|  | writer = csv.writer(response) |
|  | writer.writerow(['Ename', 'EID']) |
|  |  |
|  | employees = Employee.objects.all().values\_list('ename', 'eid') |
|  | for employee in employees: |
|  | writer.writerow(employee) |
|  |  |
|  | return response |
|  |  |
|  | 4) |
|  |  |
|  | response = input("Please enter a string message: ") |
|  |  |
|  | # Create a PDF file |
|  | from reportlab.pdfgen import canvas |
|  |  |
|  | # Create a file name |
|  | pdf\_file\_name = 'message.pdf' |
|  |  |
|  | # Create a canvas object |
|  | c = canvas.Canvas(pdf\_file\_name) |
|  |  |
|  | # Add text to the PDF |
|  | c.drawString(50, 750, response) |
|  |  |
|  | # Save the PDF |
|  | c.save() |