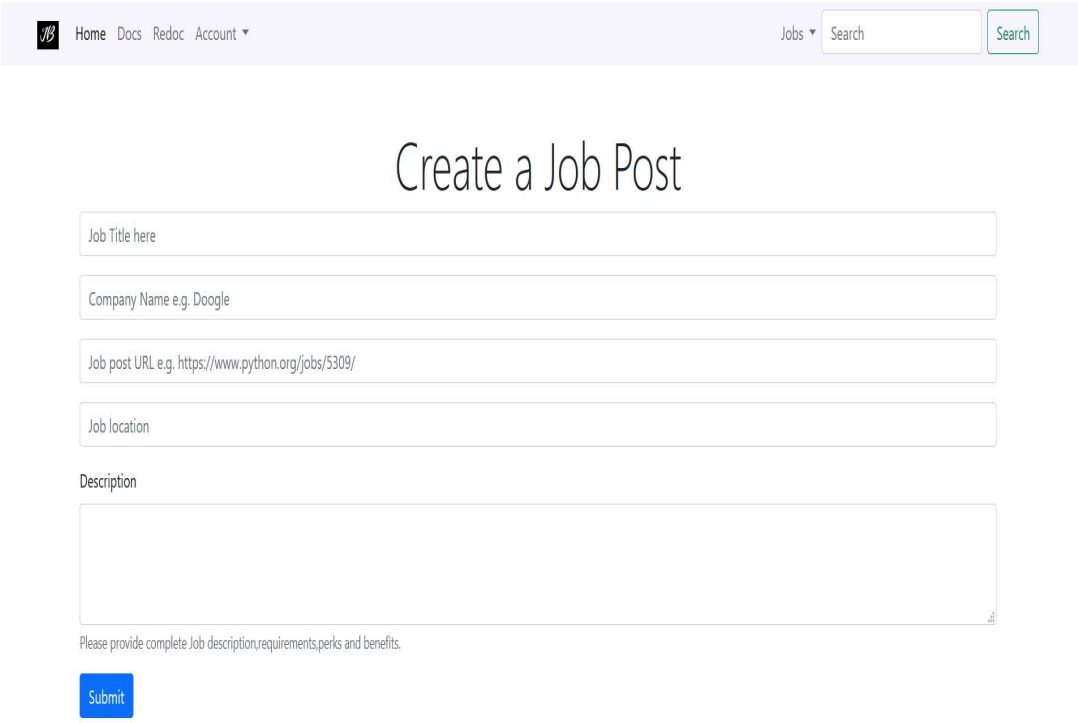
FASTAPITUTORIAL

Message!
This is an old work, for updated version please visit Updated FastAPI WebApp Course

31 : Fastapi and Jinja to Create a Job Post

Updated Udemy Course | Updated WebApp Course

We are going to implement a feature that will allow our admins and website users to create a Job post. We want a form like this which our users will fill and afterward will be redirected to the detail page of the job post.



We need a way to validate the form, again I am going to use the same technique of creating a class and validating the POST request in its

methods. We will pass the Request to initialize an object of the class and later load data into its attributes and validate the data. You might think, why I am not using Pydantic classes? It's because Pydantic classes have a default behavior to raise exceptions whenever anything is wrong. Lets create a new file **webapps** > **jobs** > **forms.py** and put the below code in it.

Copy

```
from typing import List
from typing import Optional

from fastapi import Request


class JobCreateForm:
    def __init__(self, request: Request):
        self.request: Request = request
        self.errors: List = []
        self.title: Optional[str] = None
        self.company: Optional[str] = None
        self.company_url: Optional[str] = None
        self.location: Optional[str] = None
        self.description: Optional[str] = None

    async def load_data(self):
        form = await self.request.form()
        self.title = form.get("title")
        self.company = form.get("company")
        self.company_url = form.get("company_url")
        self.location = form.get("location")
        self.description = form.get("description")

    def is_valid(self):
        if not self.title or not len(self.title) >= 4:
            self.errors.append("A valid title is required")
        if not self.company_url or not (self.company_url.__contair
            self.errors.append("Valid Url is required e.g. https:/
        if not self.company or not len(self.company) >= 1:
            self.errors.append("A valid company is required")
        if not self.description or not len(self.description) >= 2(
```

```
            self.errors.append("Description too short")
        if not self.errors:
            return True
        return False
```

Now, we can use the form in our POST request, Lets design a function that provides a form in GET request and should accept, validate and save data if POST request. Type the following code in **webapps > jobs > route_jobs.py**

Copy

```
#new additional imports
from db.models.users import User
from apis.version1.route_login import get_current_user_from_token
from webapps.jobs.forms import JobCreateForm
from schemas.jobs import JobCreate
from db.repository.jobs import create_new_job
from fastapi import responses, status
from fastapi.security.utils import get_authorization_scheme_param

#....

@router.get("/post-a-job/")       #new
def create_job(request: Request, db: Session = Depends(get_db)):
    return templates.TemplateResponse("jobs/create_job.html", {"re


@router.post("/post-a-job/")    #new
async def create_job(request: Request, db: Session = Depends(get_c
    form = JobCreateForm(request)
    await form.load_data()
    if form.is_valid():
        try:
            token = request.cookies.get("access_token")
            scheme, param = get_authorization_scheme_param(
                token
            )  # scheme will hold "Bearer" and param will hold act
            current_user: User = get_current_user_from_token(token
            job = JobCreate(**form.__dict__)
            job = create_new_job(job=job, db=db, owner_id=current_
```

```
            return responses.RedirectResponse(
                f"/details/{job.id}", status_code=status.HTTP_302_
            )
        except Exception as e:
            print(e)
            form.__dict__.get("errors").append(
                "You might not be logged in, In case problem persi
            )
            return templates.TemplateResponse("jobs/create_job.htm
    return templates.TemplateResponse("jobs/create_job.html", form
```

Now, you might have several questions, Why we are extracting token from cookies here? What is this scheme and param? It's just that we need access to the current user to create a job post, It's because every job post has an owner field that is foreign-key to a user.

Ok, then why don't we do this?

Copy

```
async def create_job(request: Request,
        db: Session = Depends(get_db),
        current_user:User = Depends(get_current_user_from_token)):
```

Definitely, this code is much cleaner and readable and gives us the current user. But only if the user is logged in and the token has not expired, Otherwise it raises an exception and our website users will see a JsonResponse "Not Authenticated" like this.

We don't want such error messages, without a UI. That's why I thought maybe we can extract the token by ourselves and pass the token to **`get_current_user_from_token`** function and get the current user manually.

Yup, it made the code dirty but at least it works. We will find a way to fix it later.

We don't need to register these 2 functions to main.py > app because the router is already registered in **webapps > base.py.**

The backend portion is done, Now moving to the frontend aspect. Create a new file **templates > jobs > create_job.html** and paste the below template code.

Copy

```
{% extends "shared/base.html" %}


{% block title %}
  <title>Create a Job Post</title>
{% endblock %}

{% block content %}
  <div class="container">
    <div class="row">
      <div class="text-danger font-weight-bold">
        {% for error in errors %}
          <li>{{error}}</li>
        {% endfor %}
      </div>
    </div>

    <div class="row my-5">
      <h3 class="text-center display-4">Create a Job Post</h3>
      <form method="POST">
        <div class="mb-3">
          <input type="text" required class="form-control" name="t
        </div>
        <div class="mb-3">
          <input type="text" required placeholder="Company Name e.
        </div>
        <div class="mb-3">
          <input type="text" required placeholder="Job post URL e.
        </div>
        <div class="mb-3">
```

```
            <input type="text" required placeholder="Job location" r
          </div>
          <div class="mb-3">
            <label for="exampleFormControlTextarea1" class="form-lab
            <textarea class="form-control" required name="descriptic
            <div id="help" class="form-text">Please provide complete

          </div>
          <button type="submit" class="btn btn-primary">Submit</butt
        </form>
      </div>
    </div>
    {% endblock %}
```

Time to test our new feature: visit http://127.0.0.1:8000/post-a-job/

In case you want to hide this feature from the general public, You can simply keep this url for this page secret and not put it in the navbar. If you don't want to publish a job post instantly and review before publishing,You can make use of the **`is_active`** field on the Job Model. When someone non-admin creates a job post it can be **`False`** by default.
Git Commit : https://github.com/nofoobar/JobBoard-Fastapi/commit/bec0017a0d8ca380ef27dac52c2932d5880a6d9b

Prev: 30 : Implementing ... Next: 32 : Deleting ...