ChatGPT

Mounting Gradio in FastAPI

Gradio can indeed be embedded in a FastAPI app using the built-in helper. For example, you can write:

```
from fastapi import FastAPI
import gradio as gr

app = FastAPI()
io = gr.Interface(lambda x: "Hello, " + x + "!", "textbox", "textbox")
app = gr.mount_gradio_app(app, io, path="/gradio")
```

This hosts the Gradio interface at `http://localhost:8000/gradio` under your FastAPI server [1]. In other words, FastAPI's `mount()` (via `gr.mount_gradio_app`) attaches the Gradio Blocks/Interface to your existing app, so everything runs in one server process. This makes it easy to share the same session and routing logic.

Passing User and Session Context

Within the mounted Gradio app, you can access the FastAPI request and session data. Any Gradio callback function can accept a parameter of type `gr.Request`, which wraps the underlying FastAPI `Request` [2]. For example:

```
def chat_fn(message, history, request: gr.Request):
    user_id = request.username  # if using Gradio auth
    session_hash = request.session_hash
    # or access full FastAPI request via request.request
    username = request.request.session.get("username")
    ...
    return response, updated_history
```

Here, `request.request` is the actual FastAPI `Request` object. If you have added a `SessionMiddleware` to the FastAPI app, you can retrieve `request.request.session[...]` just as you would in FastAPI [3]. (In one example, a callback used `request.request.session.get('username')` to show a welcome message [3].) Likewise, if you set up an `auth_dependency` in `mount_gradio_app`, Gradio will enforce your FastAPI auth logic and populate `request.username` accordingly [4] [5].

You can also use Gradio's own **state** or hidden components to pass context. For instance, a `gr.State` component can hold per-session data (initialized on load via `demo.load`) and be passed as an extra argument to your function. For example, one can call `demo.load(get_user_info,` `outputs=user_info_html)` on page load to fetch FastAPI session info and populate a hidden HTML or State component [6]. In short, any user or session data managed in FastAPI can be accessed in the Gradio callbacks via `gr.Request` or preloaded into Gradio state.

Handling Redirects and Events

Gradio's Python API does **not** currently provide a built-in way for a callback to change the browser's URL. Instead, you must use client-side JavaScript or FastAPI redirects. For example, you can place a button or link in your Gradio app that, when clicked, executes JS to change `window.location`. Gradio supports adding a JavaScript snippet to an event via the `js` or `_js` argument. For example:

```
button = gr.Button("Go to Home")
button.click(None, [], [], _js="window.location.pathname='/home'")
```

When this button is clicked, the browser navigates to `/home` [7]. Similarly, as one GitHub comment shows, you could do:

```
greet_btn.click(None, None, None, js="() => {window.location.href = '/up'}")
```

to redirect to `/up` [8]. In practice, you can also define a FastAPI route that returns a `RedirectResponse` (as the parent app) and simply link to it in the Gradio UI.

In summary: you **can** capture events in Gradio (e.g. button clicks) and perform a redirect, but this must be done with front-end JS or by invoking a separate FastAPI endpoint. There is currently no Gradio callback that returns a redirect directly; you would handle it either by JavaScript (via `_js` or `js` as shown above) or by having your FastAPI side routes serve the redirect.

Conclusion

Yes – by switching from Django to FastAPI you can mount your Gradio app in the same server and share context. Use `gr.mount_gradio_app(...)` to attach Gradio, and in your Gradio functions include a `request: gr.Request` parameter. Through that (and FastAPI's SessionMiddleware or `auth_dependency`) you can access user IDs or session state inside the Gradio callbacks [2] [3]. For redirecting the user after a Gradio event, inject JavaScript via the event listeners (`js` or `_js`) or have the Gradio UI invoke a FastAPI redirect route [9] [10] [8].

Sources: Gradio's documentation and issue discussions show how to mount onto FastAPI and use `gr.Request` for accessing headers, cookies, and session data [1] [2]. They also explain that redirecting to a new page is done via client-side JS (e.g. `_js="window.location..."`) rather than a return value in Python [9] [8]. The StackOverflow answer demonstrates using `demo.load()` to initialize session-based content on load [6].

---

[1] [4] Gradio Docs
https://www.gradio.app/docs/gradio/mount_gradio_app

[2] [5] Gradio Docs
https://www.gradio.app/docs/gradio/request

[3] [6] python - Gradio HTML component display mounted on FAST API - Stack Overflow
https://stackoverflow.com/questions/77195870/gradio-html-component-display-mounted-on-fast-api

[7] [9] [10] Embedding gradio within a fastapi app · Issue #1608 · gradio-app/gradio · GitHub
https://github.com/gradio-app/gradio/issues/1608

[8] Allow event listeners to redirect to another page · Issue #7838 · gradio-app/gradio · GitHub
https://github.com/gradio-app/gradio/issues/7838