Path Params, Query Params, and HTTPException Explained

This document breaks down the key concepts demonstrated in the parameters_and_exceptions_api.py file.

1. Path Parameters

A Path Parameter is a required part of the URL path itself. It's used to identify a specific resource.

- **In our code:** @app.get("/items/{item_id}")
- The {item_id} part is a placeholder for a value. The value provided in the URL (e.g., the 1 in /items/1) is passed to the item_id argument of our function.
- **Purpose:** You use path parameters when the API needs that value to find the exact thing the user is asking for. You cannot get an item *without* its ID.
- **Validation:** By using type hints (item_id: int), FastAPI automatically validates that the value in the URL is an integer and converts it for you. If you typed /items/abc, FastAPI would automatically return a 422 Unprocessable Entity error.

2. Query Parameters

Query Parameters are optional key-value pairs that come after a ? in the URL. They are used to modify the request, like sorting, filtering, or controlling the response format.

- **In our code:** The brand: Optional[str] = None and show_category: bool = True arguments in our function are query parameters.
- **Purpose:** They are not for identifying the resource, but for changing *how* you get it or *what* information you get back. A request for /items/2 and /items/2?brand=ErgoMax are both asking for the same item (ID 2), but the second request provides extra, optional information.
- **Syntax:**
  - In the URL: ?key=value. Multiple parameters are separated by &. Example: .../items/2?brand=ErgoMax&show_category=true
  - In the code: Any function argument that is not part of the path is treated as a query parameter.
  - Optional[str] = None: Makes the parameter optional. If the user doesn't provide it, its value will be None.
  - bool = True: You can set any default value. The user can override it in the URL (e.g., ?show_category=false).

3. Handling Errors with HTTPException

When something goes wrong (like a user asking for an item that doesn't exist), you shouldn't just crash or return a simple text message. You need to send a proper HTTP error response. HTTPException is FastAPI's way to do this.

- **In our code:**
  ```
  if item_id not in inventory_db:
      raise HTTPException(
          status_code=404,
          detail="Item with ID... not found."
      )
  ```

- **Why use it?** Raising an HTTPException tells FastAPI to stop executing the function and immediately send an error response to the client.
- **Components:**
  - status_code: This is the most important part. A **404 Not Found** status code is the standard way to tell a client that the resource they asked for doesn't exist. Browsers and other applications understand these codes.
  - detail: This provides a human-readable message in the JSON response body, explaining what went wrong.

Without HTTPException, our app would crash with a KeyError. With it, we send a clean, professional, and standard API error response.