

Making an API

GET by id

- uses path parameter

```
@app.get("/items/{item_id}")  
async def read_item(item_id:str) -> Item | str:  
    return f"item with id {item_id}"
```

```
r = requests.get(f'{url}/items/abc123')
```

GET all

- I'm giving the str option for return here just for demo purposes

```
# server.py
@app.get('/items')
async def read_items() -> list[Item] | str:
    return "all items"
```

```
# client.py
r = requests.get(f'{url}/items')
```

GET by category

- now we will modify `read_items` to allow for a query
- uses query parameter
- handles get all and get by category.

```
@app.get('/items')
async def read_items(category:str =None) -> list[Item] | str:
    if category:
        return f'items in category {category}'
    else:
        return "all items"
```

```
r = requests.get(f'{url}/items?category=office')
```

More complex queries

- If there's a complex object, we might want to handle many query parameters
- We can access the request object directly
- If we want to, we can validate the query by turning it into an ItemQuery object (assuming we have defined that model)
- IMPORTANT: When we dump the query, we will need to specify `exclude_none=True`

More complex queries

```
from fastapi import Request
# ---
@app.get('/items')
async def read_items(req: Request) -> list[Item] | str:
    '''read users and return result'''

    query = dict(req.query_params)

    if query:
        query = ItemQuery(**query)
        return f"GET querying by {query}"

    return "GET getting all items"
```

More complex queries

- This is another way to specify query parameters

```
r = requests.get(f'{url}/items', params={  
    'name': 'stapler',  
    'category': 'office'})
```

resolves to:

```
URL/items?name=stapler&category=office
```

Updating items

- Update uses a path parameter and also a request body

```
@app.put('/items/{item_id}')
async def update_item(item_id, update: ItemUpdate) -> str:
    '''read users and return result'''

    return f"updating item {item_id} with {update}"
```

```
request_body = {'category': 'office supplies'}
r = requests.put(f'{url}/items/{item_id}', json = request_body)
```