Updating Resources

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

Updating resources

- PUT for full updates
- PATCH for partial updates

Upserting

- Creating a resource with PUT or PATCH

Demo

Updating a Resource (Part 1)

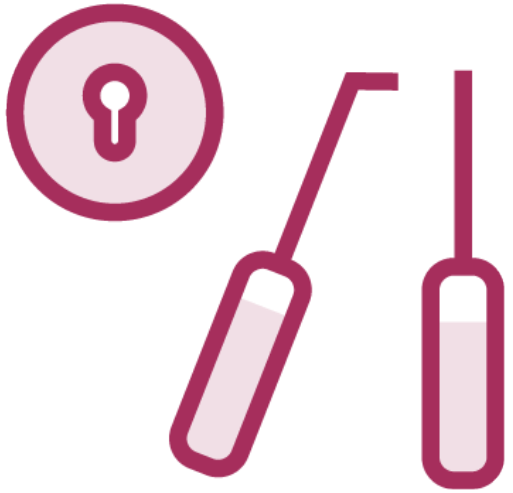
The Repository Pattern

An abstraction that reduces complexity and aims to make the code, safe for the repository implementation, persistence ignorant

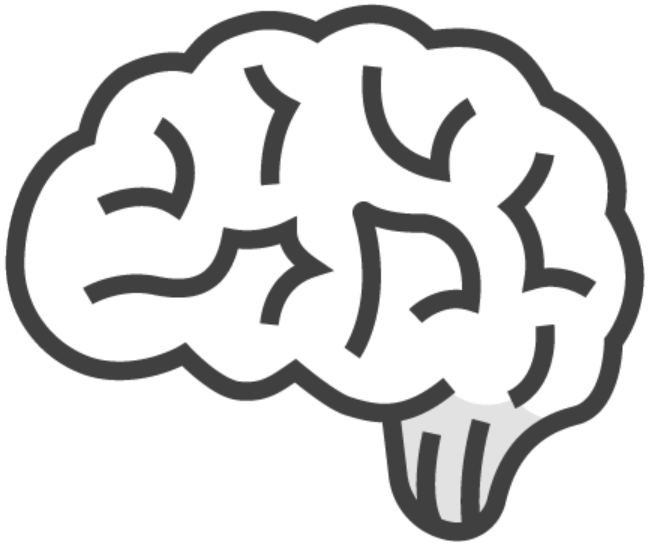
Advantages of the Repository Pattern

Less code duplication

Less error-prone code



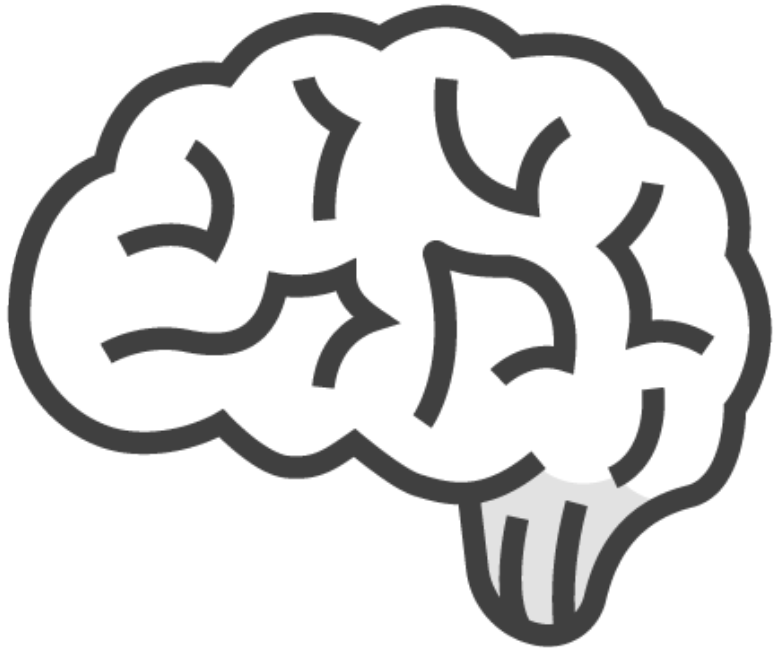
Better testability of the consuming class



Persistence ignorant

Switching out the persistence technology is not the main purpose. Choosing the best one for each repository method is.

The Repository Pattern



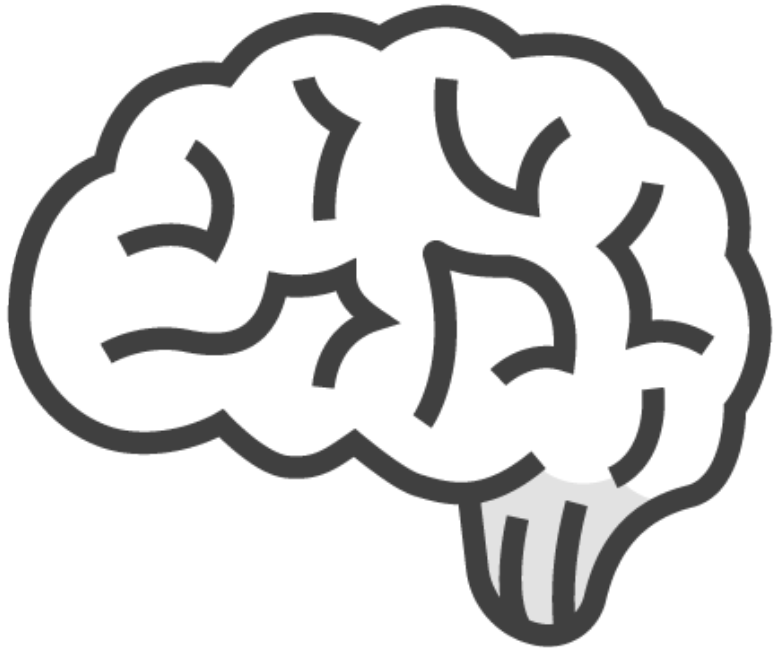
We're working on a contract, not on an implementation

Always have a set of methods matching the required functionality and call them, even if they don't do anything in the current implementation

Demo

Updating a Resource (Part 2)

Updating Collection Resources



Sending a PUT request to a collection resource like http://host/api/authors/{authorId}/books is allowed

- The books resource would be overwritten with the new collection

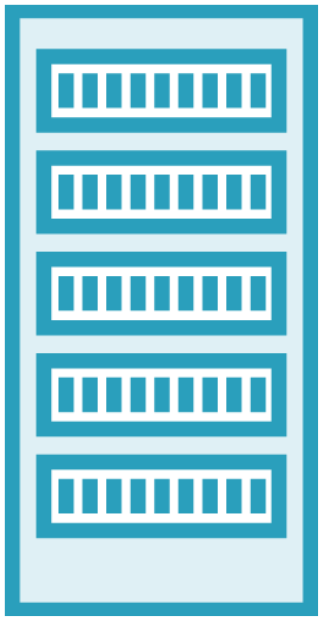
It's rarely implemented because it can be very destructive

Upserting

http://myapi/authors

http://myapi/authors/{guid}

http://myapi/authors/1



Upserting

Server is responsible for URI generation

PUT request must go to an existing URI

If it doesn't exist, a 404 is returned

POST must be used for creation as we cannot know the URI in advance

Consumer is responsible for URI generation

PUT request can be sent to an unexisting URI, because the consumer is allowed to create it

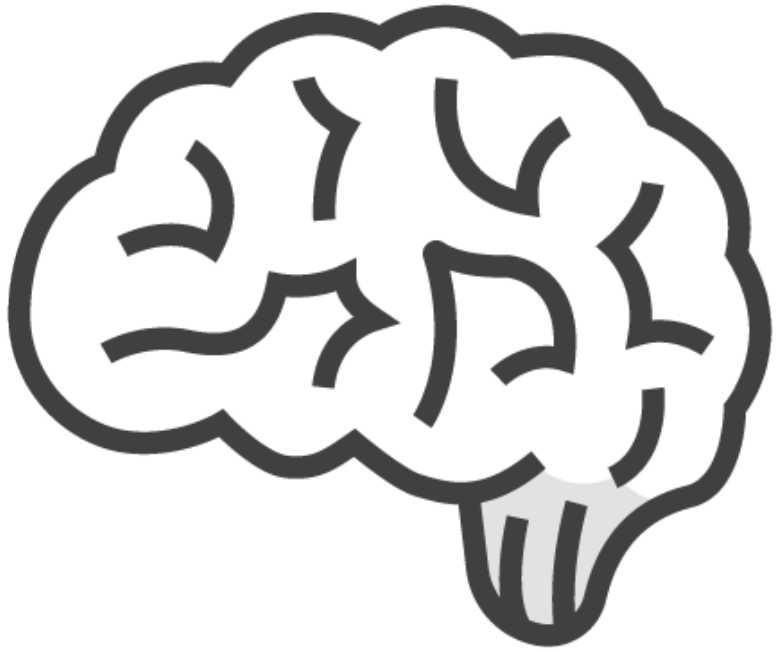
If it doesn't exist, the resource is created

PUT can be used for creation: upsert

Demo

Upserting with PUT

Partially Updating a Resource



HTTP PATCH is for partial updates

The request body of a patch request is described by RFC 6902 (JSON Patch)
https://tools.ietf.org/html/rfc6902

Patch requests should be sent with media type “application/json-patch+json”

```
[
    {
        "op": "replace",
        "path": "/title",
        "value": “new title"
    }
    ,
    {
        "op": “remove",
        "path": "/description"
    }
]
```

- array of operations
- “replace” operation
- “title” property gets value “new title”
- “remove” operation
- “description” property is removed (set to its default value)

JSON Patch Operations

Add

```
{"op": "add",
"path": "/a/b",
"value": "foo"}
```

Remove

```
{"op": "remove",
"path": "/a/b"}
```

Replace

```
{"op": "replace",
"path": "/a/b",
"value": "foo"}
```

JSON Patch Operations

Copy

```
{"op": "copy",
"from": "/a/b",
"path": "/a/c"}
```

Move

```
{"op": "move",
"from": "a/b",
"path": "/a/c"}
```

Test

```
{"op": "test",
"path": "/a/b",
"value": "foo"}
```

Demo

Partially Updating a Resource

Demo

Upserting with PATCH

HTTP Method Overview by Use Case

Reading resources

GET api/authors

200 [{author},{author}], 404

GET api/authors/{authorId}

200 {author}, 404

Deleting resources

DELETE api/authors/{authorId}

204, 404

DELETE api/authors

204, 404

Rarely implemented

HTTP Method Overview by Use Case

Creating resources (server)

POST api/authors – {author}

201 {author}, 404

POST api/authors/{authorId} can never be successful (404 or 409)

Create a new resource for adding a collection in one go

POST api/authorcollections – {authorCollection}

201 {authorCollection}, 404

Creating resources (consumer)

PUT api/authors/{authorId} - {author}

201 {author}

PATCH api/authors/{authorId} - {JsonPatchDocument on author}

201 {author}

HTTP Method Overview by Use Case

Updating resources (full)

PUT api/authors/{authorId} – {author}

200 {author}, 204, 404

PUT api/authors – [{author}, {author}]

200 [{author}, {author}], 204, 404

Rarely implemented

Updating resources (partial)

PATCH api/authors/{authorId} - {JsonPatchDocument on author}

200 {author}, 204, 404

PATCH api/authors – {JsonPatchDocument on authors}

200 [{author}, {author}], 204, 404

Rarely implemented

Summary

PUT for full updates

- 200 Ok or 204 No content
- Not safe
- Idempotent

PATCH for partial updates

- JSON Patch standard
- 200 Ok or 204 No content
- Not safe
- Not idempotent

Summary

Upserting

- Possible when the consumer of the API is allowed to create the resource URI