”Clean Architecture in C#”

Lab Manual

Wincubate ApS

19-05-2024

V1.0

Table of Contents

Exercise types

The exercises in the present lab manual differs in type and difficulty. Most exercises can be solved by applying the techniques from the presentations in the slides in a more or less direct manner. Such exercises are not categorized further.

However, the remaining exercises differs slightly in the sense that they are not necessarily easily solvable. These are categorized as follows:

★	Labs marked with a single star denote that the corresponding exercises are a bit more loosely specified.
★★	Labs marked with two stars denote that the corresponding exercises contain only a few hints (or none at all!) or might be a bit more difficult or nonessential. They might even require additional searches for information elsewhere than in the slide presentations.
★★★	Labs marked with three stars denote that the corresponding exercises are not expected in any way to be solved. These are difficult, tricky, or mind-bending exercises for the interested participants – mostly for fun! ☺

Prerequisites

The present labs require the course files accompanying the course to be extracted in some directory path, e.g.

```
C:\Wincubate\CleanArchitecture
```

with Visual Studio 2022 with .NET 8 installed on the PC.

We will henceforth refer to the chosen installation path containing the lab files as *PathToCourseFiles* .

Module 3: “Our Running Example”

Lab 03.1: ”Add a System Clock to ShortR” (★)

This lab investigates where to put cross-cutting concerns such as time in the Clean Architecture structure.

- Open the starter project in
  PathToCourseFiles`\Modules\03 – Our Running Example\Labs\Lab 03.1\Begin`

which contains ShortR as it (roughly) looked at the end of the presentation for Module 03.

In **Application** we have made a somewhat nasty error by calling `DateTime.Now` directly when time stamping a newly created `ShortenedUrl`.

- Is this really a problem? Explain why.

It is now your task to implement a better version of the time stamping code.

- Implement appropriate interface(s) and class(es) to capture the system clock to fix the above problem in a proper Clean Architecture manner.
  - Which layers do the appropriate types belong to? And why…?
  - Which access modifiers have you given your types?

Module 4: “CQRS and Mediator”

Lab 04.1: ”Get all shortened URLs” (⭐⭐)

This lab extends the two use cases **Shorten** and **Goto** with a third use case.

- Open the starter project in
  PathToCourseFiles`\Modules\04 – CQRS and Mediator\Labs\Lab 04.1\Begin` ,

which contains ShortR as it (roughly) looked at the end of the presentation for Module 04.

The use case you will implement in this lab allows the user to retrieve a condensed view of all the shortened URLs in the service. Let’s refer to this new functionality as **GetUrls**.

- Inspect the `ShortR.Api.http` file in the project which you will use for testing in this exercise.

This file has been extended to include a request for **GetUrls** as follows:

```
GET https://localhost:7044/shortenedurls
```

When this request is executed towards the service after having executed all the other requests in the file, it will return a `200 OK` response result similar to:

```
{
  "urls": [
    {
      "id": "be41769b-fc24-4626-a8a3-73bd45ad3274",
      "code": "rammstein",
      "url": "https://www.rammstein.de/de/konzerte/gelsenkirchen-27-07-
2024/"
    },
    {
      "id": "fe5e74e4-fc4a-46b7-92c2-13587264a879",
      "code": "swiftie",
      "url": "https://www.fansale.de/tickets/all/taylor-swift/448410"
    }
  ]
}
```

Of course, the ids may vary.

- Implement the **GetUrls** use case in the ShortR service architecture of Module 04.

<u>Note:</u> A few hints to completing the lab:

1. To implement **GetUrls** you will need to extend in the appropriate places in all layers – except **Domain**.
2. There may be some degenerate cases along the way: Empty parts of queries, mappers, or similar.

If Time Permits: Lab 04.2: ”Should DTOs be decorated?” (⭐)

This lab provides an opinionated technical discussion about DTOs in and out of services.

- Open the starter project in
  PathToCourseFiles`\Modules\04 – CQRS and Mediator\Labs\Lab 04.2\Begin` ,

which contains a completed solution to Lab 04.1 above (and includes all DTOs from presentation as well).

- Would you have decorated the DTOs any further? Why (not)?

Module 5: “Error Handling”

If Time Permits: Lab 05.1: ”Delete a shortened URL” (⭐⭐)

This lab once again extends the set of use cases.

- Open the starter project in
  PathToCourseFiles`\Modules\05 – Error Handling\Labs\Lab 05.1\Begin` ,

which contains ShortR as it (roughly) looked at the end of the presentation for Module 05.

The use case you will implement in this lab allows the user to delete a shortened URLs by Id. Let’s refer to this new functionality as **Delete**.

- Inspect the `ShortR.Api.http` file in the project which you will use for testing in this exercise.

This file has been extended to include a request for **Delete** as follows:

```
DELETE https://localhost:7044/shortenedurls/be41769b-fc24-4626-a8a3-73bd45ad3274
```

When this request is executed towards the service, it will return a

i. `204 NoContent` response with empty body, if successfully deleted
ii. `404 NotFound` response, if no such shortened URL exists

Your task is now:

- Implement the **Delete** use case in the ShortR service architecture of Module 05.
  - Don’t worry about concurrency issues in the repository!

Module 6: “Validation”

Lab 06.1: ”Add validation to Goto”

This lab investigates the error handling and validation of Module 05 and 06.

- Open the starter project in
  PathToCourseFiles`\Modules\06 – Validation\Labs\Lab 06.1\Begin` ,

which contains ShortR as it (roughly) looked at the end of the presentation for Module 06.

Your task is to add validation also to the **Goto** use case following the existing structure we have just developed in Module 06.

- Implement validation for **Goto** such that the code property is ensured to be
  - between 5 and 30 characters
  - characters must be alphanumeric (letters or digits)