Темы урока

На самом деле на этом уроке мы просто доделываем то, что мы начали на прошлом. Презентация практически та же (но откопирована на случай, если потребуется внести какие-нибудь уточнения или изменения), указания, в принципе, все те же.

Организация слоя данных чат-бота как отдельного сервиса Web API

Постановка задачи

Что мы хотим сделать?

Мы хотим обеспечить сервисный доступ к списку напоминаний через Web API. Т.е., иметь возможность смотреть список существующих напоминаний, добавлять новые напоминания, удалять ещё не завершенные напоминания.

Как мы будем это делать?

Если бы у нас была реальная база данных (т.е. уже был бы выделен единственный сервис с абсолютной адресацией), мы могли бы написать стороннее Web API приложение, которое бы ссылалось на `Reminder.Storage.Core` и `Reminder.Storage.Sql`, например. Иными словами, мы

смогли бы себе позволить иметь сборку `Reminder.Storage.Sql` в двух независимых проектах, так как она просто обеспечивает доступ к базе данных. База при этом единственна.

Однако, у нас есть просто `Reminder.Storage.InMemory`, данные хранятся непосредственно в сборке `Reminder.Storage.InMemory`, а значит, если мы её продублируем для Web API, у него будет просто свой собственный набор напоминаний, никак не связанных с данными реального приложения чат-бота.

Поэтому нам придется обеспечить единственность `Reminder.Storage.InMemory` и выставлять её через Web API для как для пользователей Web API, так и для приложения.

Разработка

Продолжаем двигаться итеративно метод за методом:

- Сначала метод в Web API
- Затем метод, его использующий в библиотеке клиента Web API
- Смотрим на результат в наскоро набросанном консольном приложении для тестирования клиента (и только в самом конце мигрируем в приложении телеграм-бота)
- Затем переходим к следующему методу по списку в порядке, указанном ниже

Стремиться нужно к результату тут:
https://github.com/ago-cs/cs-course-q3/tree/master/Lessons/25/ClassWork/Final.

Вся работа у меня заняла плотных 2 урока (8 часов) в режиме “я пишу и объясняю, они пишут у себя”.

Reminder.Storage.WebApi

Эта сборка — Web API к нашему хранилищу.

Здесь всё должно идти гладко, так как именно это мы и делали на прошлом занятии.

Рекомендуемая последовательность реализации методов:

- ```
  [HttpGet("{id}", Name = "GetReminder")]
  public IActionResult GetReminder(Guid id)
  ```
  * он не столько нужен функционально, однако он потребуется для Web API как часть ответа 201 - Created (там возвращается заголовок Location, который имеет значение URL, по которому доступен вновь созданный ресурс).
- ```
  [HttpPost]
  public IActionResult CreateReminder(
        [FromBody] ReminderItemCreateModel reminder);
  ```

- ```
  [HttpGet]
  List<ReminderItem> GetReminder(
        [FromQuery(Name = "[filter]status")] int status = -1,);
  ```
- ```
  [HttpPatch("{id}")]
  void UpdateReminderStatus(
        Guid id,
        [FromBody] JsonPatchDocument<ReminderItemUpdateModel> patchDocument);
  ```
- ```
  [HttpPatch]
  void UpdateRemindesrStatus(
        [FromBody] ReminderItemsUpdateModel reminderItemsUpdateModel);
  ```

Reminder.Storage.WebApi.Client

Эта сборка — клиентская библиотека упрощающая работу с нашим Web API.

Reminder.Storage.WebApi.Core

Сюда в процессе написания перенесётся общий код между первыми двумя сборками.

Возможности для обсуждения архитектуры

На самом деле, вопрос спорный, нужно ли выделять модели контроллера нашего Web API (`Reminder.Storage.WebApi`) в отдельную сборку.

В данном конкретном случае, библиотека клиента `Reminder.Storage.WebApi.Client` ссылается 2 библиотеки:

- `Reminder.Storage.WebApi.Core`
- `Reminder.Storage.Core`

Если разместить классы моделей контроллера `Reminder.Storage.WebApi` в библиотеке `Reminder.Storage.Core`. Это уменьшило бы количество библиотек, которые необходимо “тащить за собой” клиенту. Однако, самому приложению `Reminder.App` классы моделей контроллера `Reminder.Storage.WebApi` не нужны.

В связи с этим, я решил вынести классы моделей контроллера `Reminder.Storage.WebApi` в библиотеку `Reminder.Storage.WebApi.Core`. Однако, надо понимать, что “правильного” ответа здесь нет, и большой проблемы не будет, если все служебные классы, которые нужны более, чем одной внешней сборке, объединить в `Reminder.Storage.Core`.

Домашнее задание

Раскомментировать временно закомментированные методы `IReminderStorage` и поддержать их и Web API и в клиентской библиотеке..
  ```

Я давал его как “Д/з без оценки”, чтобы попытались сами доделать. Это скорее для саморазвития. В качестве примера реализации я выложил им в домашку пример выполнения https://github.com/ago-cs/cs-course-project2.