

Документация Clerk

Актуально на 1.1.5 (MyBad!)

Компоненты

Документация

Rukn.Data

Rukn.Data

Главное

Набор базовых интерфейсов и их реализаций для обмена данными между библиотеками и другими видами программного обеспечения.

Структура

- **Abstractions** - интерфейсы для обмена информацией между ПО.
 - **Models** - для моделей.
 - **ILesson** - занятие.
 - **IHasRelevance** - с актуальностью.
 - **IHasTimeRange** - с временем начала и конца.
 - **Providers** - для поставщиков данных.
 - **ILessonsProvider** - поставщик занятий.
 - **IDayLessonsProvider** - поставщик занятий на день.
 - **IWeekLessonsProvider** - поставщик занятий на неделю.
 - **IScheduleProvider** - поставщик расписания занятий (*IDayLessonsProvider* и *IDayLessonsProvider* и *IWeekLessonsProvider*).
- **Models** - простые реализации интерфейсов.
 - **Lesson** - простая реализация интерфейса занятия **ILesson**.
 - **RelevanceLesson** - **Lesson** с реализацией **IHasRelevance**.
 - **TimedLesson** - **Lesson** с реализацией **IHasTimeRange**.
 - **TimedRelevanceLesson** - **Lesson** с реализацией **IHasRelevance** и **IHasTimeRange**.
 - **ScheduleProviderContainer** - контейнер для **IDayLessonsProvider** и **IWeekLessonsProvider** с реализацией **IScheduleProvider**.

Rukn.Data.Pretty

Rukn.Data.Pretty

Главное

Красивое текстовое представление данных.

Немного информации

- Множество элементов обрабатывает компоненты занятия, или требуют реализации этого.
- Механизм **MultiLesson** нацелен группировку занятий с повторяющимися компонентам.
- **PrettyLesson** - реализация интерфейса **ILesson** из **Rukn.Data** с заменой ToString.
- **PrettySettings** - настройки отображения. (Рекомендуется создать набор, желательно неизменяемых настроек, для переиспользования)

RucSu

RucSu

Главное

RucSu - поставщик данных РУК на основе обработке HTML с помощью Regex, или использования Json API.

Принцип работы RegexParser

1. Получает HTML код путем POST запроса с определенными параметрами.
2. С помощью регулярных выражений вырезает из HTML данные о днях и делит на отдельные единицы.
3. С помощью регулярных выражений делит дни на занятия.
4. С помощью регулярных выражений делит занятия на: номер предмета, название предмета, имя преподавателя или название группы, и место, где будет производиться занятия.
5. Создаёт экземпляры соответствующих классов и заполняет их нужными данными.

Тонкости

Регулярные выражения

Получение дня

```
"bold\">\\s+(.*?)\\s\\(.*?)</div>\\s+</div>"
```

Получение занятия

```
"([0-9]*)\\. (.*?)<.*?/>\\s+(.*?)<br/>\\s+(.*?),\\s+(.*?)<"
```

Получение опций

```
"value=\"(.*?)\".*?>(.*?)</option>"
```

Получение выборки

```
"lg\" name=\"(.*?)\".*?>(.*?)</select>"
```

Параметры *post* запроса

Имя параметра	описание	пример значения
branch	филиал	смотреть на сайте
year	год	смотреть на сайте
group	группа	смотреть на сайте
search-date	поиск	search-date
date-search	ввод даты для поиска.	2024-05-20

Пример параметров

```
"branch=a3a10303-2b2d-4490-bccf-9d6ffc3b415d&year=46b0ccd1-5efc-40c5-bb72-1d002428937b&group=3a1d8858-59ee-4c21-b4c0-560592d7ce8d&search-date=search-date"
```

Структура

- **Models** (содержит профиль, и жестко закодированное время расписания занятий, что значит получаемое не с сайта)
- **Services** (сервисы)

- **Parsers** (получения данных с сайта, их обработка)
- **Wrappers** (упрощённая оболочка над **RegexParser**)
- **RucSuLessonsProvider** (реализация **IScheduleProvider** из **Rukn.Data**)

RucSu.DataStore

RucSu.DataStore

Главное

Интерфейс хранилища данных для поставщика **RucSu**.

Внимание, не является реализацией, а требует реализацию хранилища.

RucSu.DataStore.EasyDB

RucSu.DataStore.EasyDB

Реализация хранилища данных поставщика RucSu на основе **ADO.NET** с обёрткой **EasyDB**.

RucSu.DataStore.EFC

RucSu.DataStore.EFC

Реализация хранилища данных поставщика RucSu на основе **Entity Framework Core**.

RucSu.Tools

RucSu.Tools

Набор полезных вещей для RucSu.

About

Clerk

Это дочерний проект основного проекта [Rukn](#), представляющий собой графическое приложение для мобильных платформ.

Поставщик: shadowsystem .

Сценарии (скрипты)

ОСНОВЫ

В качестве языка для скриптов используется JavaScript (далее JS).

Для просмотра доступного API я рекомендую использовать рефлексия.

Например для просмотра доступных объектов можно использовать следующий код:

```
clear();
for (key in this)
    log(key);
```

Но этого недостаточно, так как мы узнаём только о доступных объектах. Поэтому есть класс Reflection, который включает несколько методов:

Для просмотра методов (функций) объекта можно использовать следующий код:

```
clear();
let data = Reflection.GetMethods(объект);
for (let key in data)
    log(`${data[key]} ${key}`);
```

Где объект - сам объект (для экземпляров классов), или строка с его наименованием (для классов и типов). Например для самого объекта рефлексии:

```
clear();
let data = Reflection.GetMethods(Reflection);
for (let key in data)
    log(`${data[key]} ${key}`);
```

Здесь вы увидите другие доступные методы для рефлексии, например просмотра свойств, полей, конструкторов.

А для класса или типа надо добавить кавычки, а точнее передать строку с наименованием класса или типа:

```
clear();
let data = Reflection.GetMethods('Theme');
for (let key in data)
    log(`${data[key]} ${key}`);
```

Если функция возвращает Task или имеет приписку Async, то она выполняется асинхронно и для работы с ней используются Promise, async, await. Например:

```
(async () => { // await должен быть в асинхронных функциях
    let result await confirm("You ready?"); // здесь получаем данные от
```

```
асинхронной функции  
})).catch(log);
```

Для запуска удалённого скрипта (где URL ссылка на удалённый скрипт):

```
fetch('URL').then(eval).catch(log);
```

Примеры и публичные скрипты здесь:

<https://github.com/shadowsystemss/Update/tree/main/clerk/scripts>

На данный момент доступны:

Прямоком из CLR (.NET)

- Guid (<https://learn.microsoft.com/ru-ru/dotnet/api/system.guid>)
- DateTime (<https://learn.microsoft.com/ru-ru/dotnet/api/system.datetime>)
- TimeSpan (<https://learn.microsoft.com/ru-ru/dotnet/api/system.timespan>)
- CancellationToken (<https://learn.microsoft.com/ru-ru/dotnet/api/system.threading.cancellationtoken>)
- CancellationTokenSource (<https://learn.microsoft.com/ru-ru/dotnet/api/system.threading.cancellationtokensource>)
- Task (<https://learn.microsoft.com/ru-ru/dotnet/api/system.threading.tasks.task>)
- Path (<https://learn.microsoft.com/ru-ru/dotnet/api/system.io.path>)

Rukn (ядро)

- Lesson (модель занятия)
- RelevanceLesson (занятие с актуальностью)
- TimedLesson (занятие с временем)
- TimedRelevanceLesson (занятие с временем и актуальностью)
- MultiLesson (мультизанятие из нескольких занятий)
- PrettyLesson (украшенное занятие)
- PrettySettings (настройки украшения)
- MultiLessonsBuilder (сервис преобразования занятий в мультизанятия)

RucSu (провайдер РУК-а)

- Profile (профиль из Guid)
- FullProfile (профиль с наименованиями и Guid-ами)
- EasyProfileManager (модель для быстрого профиля)

Clerk (приложение)

- AdvertisementService (реклама)

- `PopupService` (сообщения пользователю)
- `PersonalizationService` (сервис персонализации)
- `Directory` (директории|папки)
- `File` (файлы)
- `FileSystem` (файловая система)
- `Settings` (настройки)
- `Theme` (тема)

Services (сервис)

- `Clerk` (приложение)
- `RucSu` (РУК)
- `Tools` (инструменты)
- `Rukn` (ядро)
- `Init` (инициализатор)
- `http` (`HttpClient` для доступа в сеть)

Работа с терминалом

`clear();` - очистить терминал.

`log(object);` - вывести объект в терминал.

`setCode(text);` - заменить код.

`changeMode();` - сменить режим на вывод и наоборот.

`loadDefault();` - загрузить код по умолчанию.

Код по умолчанию находится по пути `Path.Combine(FileSystem.ScriptDirectory, "default")`.

Пример:

```
clear();
log('Hello world');
changeMode();
```

Всплывающее сообщения

Функции находятся в `PopupService`.

`alert(string);` - вывести предупреждение с текстом.

`confirm(string);` - вывести вопрос с вариантами да или нет.

`prompt(string);` - вывести вопрос с полем ввода.

`select(object[]);` - вывести список с выбором.

Пример:

```
clear();
alert('Hello world');
let result = await confirm('You are fine?');
```

```
let answer = await prompt('What is your name?');  
let selection = await select(["first", "second", "third"]);
```

Настройки (Settings)

Текущий профиль (SelectedProfileId)

Строковый идентификатор активного профиля, выбранного в выпадающем списке. Используется для переключения пользовательских или системных конфигураций.

Пример:

```
Settings.SelectedProfileId = "base";  
log(Settings.SelectedProfileId);
```

Только СУБД (DbOnly)

Определяет, следует ли загружать данные исключительно из локальной базы, без запросов к серверу.

Тип: bool

true — использовать только данные СУБД.

false — разрешить сетевые запросы.

Пример:

```
Settings.DbOnly = true; // Включить оптимистичный кеш
```

Переключатель загрузчика (DownloadSwitch)

Выбор между старым механизмом загрузки (Regex) и новым (JSON API).

Тип: bool

true — использовать JSON API.

false — использовать Regex-шаблоны.

Пример:

```
Settings.DownloadSwitch = false; // Режим Regex  
Settings.DownloadSwitch = true; // Режим Json API
```

Время ожидания ответа сервера (DownloadTimeout)

Таймаут ожидания сетевого ответа.

Тип: TimeSpan

Пример:

```
Settings.DownloadTimeout = TimeSpan.FromSeconds(3); // Ждать 3 секунды
```

Авто-обновление данных в СУБД (CacheAutoUpdate)

Определяет, сохранять ли полученные с сервера данные в базу автоматически.

Тип: bool

true — обновлять.

false — не обновлять.

Пример:

```
Settings.CacheAutoUpdate = true; // Включить авто-обновление
```

Оптимистичный кеш

Разрешает использовать данные из локальной БД, даже если они устарели.

Тип данных: bool.

Значение: true (включить) или false (выключить)

Пример:

```
Settings.CacheOptimistic = true; // Включить оптимистичный кеш
```

Срок актуальности кеша (CacheRetention)

Промежуток времени, в течение которого кеш считается актуальным.

Тип: TimeSpan

Пример:

```
Settings.CacheRetention = TimeSpan.FromMinutes(30);
```

Обработчик (PipelineWrapper)

Включает дополнительный этап обработки данных (добавление временных меток, сортировка).

Тип данных: bool.

Значение: true (включить) или false (выключить)

Пример:

```
Settings.PipelineWrapper = true; // Включить оптимистичный кеш
```

Тема (Theme)

Для экспорта темы:

```
clear();  
log("clear()");  
for (i in Reflection.GetProperties('Theme'))
```

```
log(`Theme.${i} = '${Theme[i].replaceAll("\\", "\\\\"')}';`);
log("PersonalizationService.LoadTheme()");
log("log('Theme loaded');")
```

Тип данных практически везде - строка.

Значения везде, кроме `BackgroundImage` это цвета описанные HEX в `#AARRGGBB` или `#ARGB` или `#RGB` или `#RRGGBB`. Например:

```
Theme.ImageMask = '#A000';
```

`BackgroundImage` - это путь к картинке, либо ссылка на картину. Например для его изменения:

```
(async () => {
    clear();
    let file = await FileSystem.PickPhotoAsync();
    let path = Path.Combine(FileSystem.ScriptDirectory, "background");
    if (!file) {
        alert("Вы не выбрали картинку");
        return;
    }
    await File.CopyAsync(file, path, cancel);
    Theme.BackgroundImage = path;
    log("BackgroundImage setted");
})();
```

Приложение (Clerk)

Версия

```
log(Clerk.Version);
log(Clerk.VersionName);
log(Clerk.VersionId);
```

Методы

- `Task<string> Fetch(string url)`
Загружает содержимое по указанному URL и возвращает его как строку.
- `static Task Copy(string text)`
Копирует переданный текст в буфер обмена.
Пустые строки игнорируются.
- `static Task ShareText(string text, string title = "shared text")`
Открывает системное окно «Поделиться» и отправляет указанный текст.
Оба параметра должны быть непустыми.

ClerkScheduleScope

Наследуется от ScheduleProviderScope:

- `void ResetProfile()`
Сбрасывает сохранённый профиль, заставляя систему при следующем обращении заново получить FullProfile из провайдера.
- `Task<FullProfile?> GetFullProfileAsync(CancellationToken cancel)`
Возвращает полный профиль пользователя.
Если профиль уже был загружен ранее, возвращает закешированное значение.
- `Task<Profile?> GetProfileAsync(CancellationToken cancel)`
Возвращает базовый профиль (Profile) через вызов GetFullProfileAsync.
- `Task<IEnumerable<ILesson>?> GetDayLessonsAsync(DateTime date, CancellationToken cancel)`
Возвращает расписание уроков на указанный день для текущего профиля.
- `Task<IEnumerable<ILesson>?> GetWeekLessonsAsync(DateTime date, CancellationToken cancel)`
Возвращает расписание уроков на неделю, содержащую указанную дату.

Добавляет:

```
// Возвращают PrettySettings, требуют асинхронности.
let scope = Clerk.ClerkScheduleScope;
let ps = await scope.GetDayPrettySettingsAsync(CancellationToken);
let ps = await scope.GetWeekPrettySettingsAsync(CancellationToken);

// Возвращает List<MultiLesson>? - список мультизанятий.
let lessons = await scope.GetProcessedDayLessonsAsync(new Date('2025-09-01'), cancel); // День
let lessons = await scope.GetProcessedWeekLessonsAsync(new Date('2025-09-01'), cancel); // Неделя
```

TimeAddAndSortWrapper

Класс-обёртка над IFullProfileScheduleProvider , которая:

- добавляет временные интервалы (Start , End) к урокам без них;
- сортирует уроки по дате и времени;
- может быть включена или отключена через свойство IsEnabled .

Методы

- `static ILesson AddTime(ILesson lesson)`
Добавляет временной диапазон (Start, End) в урок, если он отсутствует.
Если урок уже реализует IHasTimeRange, возвращается без изменений.

- `IEnumerable<ILesson> Process(IEnumerable<ILesson> lessons)`
Добавляет временные интервалы ко всем урокам и сортирует коллекцию.
Если `IsEnabled == false`, возвращает исходную коллекцию без изменений.
- `Task<List<ILesson>?> GetDayScheduleAsync(FullProfile profile, DateTime date, CancellationToken cancel)`
Возвращает расписание на указанный день.
Результат проходит через обработку (`AddTime`, `Sort`).

Файловая система (FileSystem)

Поля

- `string DataDirectory`
Абсолютный путь к директории данных приложения (`FileSystem.AppDataDirectory`).
Запись в неё запрещена напрямую, если путь не попадает в область скриптов.
- `string CacheDirectory`
Абсолютный путь к директории кеша (`FileSystem.CacheDirectory`).
Чтение и запись разрешены.
- `string ScriptDirectory`
Абсолютный путь к директории `AppDataDirectory/script`.
Это единственная область внутри данных приложения, куда разрешена запись.

Методы

- `bool TestPath(string path)`
Определяет, допускается ли доступ к указанному пути.
Логика проверки:
 - Если путь находится внутри `DataDirectory`, он считается разрешённым **только** если также лежит внутри `ScriptDirectory`.
 - Пути, находящиеся за пределами `DataDirectory`, автоматически считаются разрешёнными.
- `void ThrowIfProhibited(string path)`
Вызывает `UnauthorizedAccessException`, если `TestPath` вернул `false`.
- `Task<FileResult?> PickFileAsync()`
Открывает системный диалог выбора файла и возвращает выбранный файл или `null`.
- `Task<IEnumerable<FileResult>> PickMultipleFileAsync()`
Позволяет выбрать несколько файлов через диалог выбора.
- `Task<FileResult?> PickPhotoAsync()`
Открывает диалог выбора изображения через системный медиапикер.

Файл (File)

- **static async Task<string> ReadAllTextAsync(FileResult file, CancellationToken ct)**
Асинхронно читает весь текст из переданного `FileResult`, открывая поток и считывая его полностью.
- **async Task CopyAsync(FileResult file, string destination, CancellationToken ct)**
Асинхронно копирует содержимое переданного файла в путь `destination`.
Перед записью выполняет проверку пути через `FileSystemApi.ThrowIfProhibited`, затем создаёт файл и копирует данные.
- **void WriteAllText(string path, string text)**
Создаёт или перезаписывает файл, записывая указанный текст.
Перед операцией проверяет путь через `FileSystemApi.ThrowIfProhibited`.
- **void AppendAllText(string path, string text)**
Добавляет текст в конец файла.
Перед операцией проверяет путь через `FileSystemApi.ThrowIfProhibited`.
- **void Copy(string source, string destination)**
Копирует файл из `source` в `destination`.
Перед копированием проверяет путь назначения через `FileSystemApi.ThrowIfProhibited`.
- **static void Delete(string path)**
Удаляет файл по указанному пути без дополнительных проверок.

Директории (Directory)

Методы

- **void Create(string path)**
Создаёт директорию по указанному пути.
Если она уже существует — ничего не делает.
Создаёт все недостающие родительские каталоги.
- **string? GetParent(string path)**
Возвращает абсолютный путь к родительской директории.
Если родитель отсутствует — возвращает `null`.
- **IEnumerable<string> GetDirectories(string path)**
Перечисляет директории верхнего уровня внутри указанного пути.
Возвращает имена в виде абсолютных путей.
- **IEnumerable<string> GetFiles(string path)**
Перечисляет файлы верхнего уровня внутри указанного пути.
Возвращает абсолютные пути к файлам.
- **bool Exists(string? path)**
Проверяет существование директории.
Возвращает `true`, если путь указывает на существующую директорию.

РУК (RucSu)

Свойства

- `IProfileProvider ProfileProvider`
Предоставляет доступ к провайдеру профилей.
- `IWeekScheduleProvider WeekScheduleProvider`
Предоставляет доступ к провайдеру недельного расписания.
- `IBranchesProvider BranchesProvider`
Позволяет получать список филиалов.
- `IYearsProvider YearsProvider`
Позволяет получать список учебных годов.
- `IGroupsProvider GroupsProvider`
Позволяет получать список учебных групп.
- `IEmployeesProvider EmployeesProvider`
Позволяет получать список сотрудников.
- `IOptionsProvider OptionsProvider`
Объединяет все провайдеры опций (филиалы, годы, группы, сотрудники).
- `IFullProfileProvider FullProfileProvider`
Позволяет работать с полными профилями (`FullProfile`).
- `IScheduleRelevanceProvider ScheduleRelevanceProvider`
Предоставляет доступ к данным актуальности расписания.
- `IFullProfileLessonsKeeper FullProfileLessonsKeeper`
Хранилище уроков для полного профиля.
- `IFullProfileDayScheduleProvider FullProfileDayScheduleProvider`
Позволяет получать расписание на день для полного профиля.
- `IFullProfileWeekScheduleProvider FullProfileWeekScheduleProvider`
Позволяет получать расписание на неделю для полного профиля.
- `IFullProfileScheduleProvider FullProfileScheduleProvider`
Базовый провайдер расписаний, объединяющий дневное и недельное расписание.

Инструменты (Tools)

Свойства

- `FullProfileFiller FullProfileFiller`
Сервис для автоматического заполнения недостающих данных в профиле.
- `EasyProfileManager EasyProfileManager`
Упрощённый менеджер профиля, работающий с заполнением через `FullProfileFiller` .
- `DbProfileManager DbProfileManager`
Менеджер профиля, взаимодействующий с базой данных.
- `DbScheduleProvider DbScheduleProvider`
Провайдер расписаний, получающих данные из базы данных.

- `CacheScheduleProvider` `CacheScheduleProvider`
Провайдер расписаний, работающий через кэш.
- `UpdatesManager` `UpdatesManager`
Сервис, управляющий обновлениями данных.
- `ScheduleDownloadSwitch` `ScheduleDownloadSwitch`
Компонент, управляющий процессом загрузки расписаний (включение/отключение).
- `OptionsTable` `OptionsTable`
Таблица настроек и опций.
- `UpdatesTable` `UpdatesTable`
Таблица обновлений расписаний.
- `LessonsTable` `LessonsTable`
Таблица уроков.
- `ProfilesTable` `ProfilesTable`
Таблица профилей пользователей.

FullProfileFiller

Свойства

- `bool` `SearchByName`
Определяет, выполнять ли поиск по имени, если точное совпадение ключа не найдено.

Методы

- `KeyValuePair<string, Guid>? GetPairByName(string? name, Dictionary<string, Guid>? options)`
Возвращает пару “имя–GUID” из словаря по имени.
Если `SearchByName == true`, дополнительно выполняется нестрогий поиск по совпадению имени.
- `Task<FullProfile> FillAsync(Profile profile, CancellationToken cancel)`
Заполняет недостающие данные профиля (филиал, год, сотрудник, группа), обращаясь к `IOptionsProvider`.

EasyProfileManager

Свойства

- `FullProfile` `Current`
Текущий профиль, редактируемый и используемый в запросах.
`bool` `EmployeeMode`
Определяет режим сотрудника.
При изменении обновляет состояние `Current`.
- `string?` `Branch`
Имя филиала. При изменении сбрасывает GUID филиала.

- `string? Year`
Имя учебного года. При изменении сбрасывает GUID года.
- `string? Employee`
Имя сотрудника. При изменении сбрасывает GUID сотрудника.
- `string? Group`
Имя группы. При изменении сбрасывает GUID группы.

Методы

- `Task<FullProfile?> GetFullProfileAsync(CancellationToken cancel)`
Возвращает или обновляет полный профиль.
При изменении `Current` заново заполняет его через `FullProfileFiller`.
- `Task<Profile?> GetProfileAsync(CancellationToken cancel)`
Возвращает профиль в базовом (`Profile`) формате.

Rukn

Свойства

- `IWeekLessonsProvider WeekLessonsProvider`
Провайдер получения уроков за неделю.
- `IDayLessonsProvider DayLessonsProvider`
Провайдер получения уроков за день.
- `ILessonNameFormatter LessonNameFormatter`
Форматирует имена уроков.
- `ITypeNameFormatter TypeNameFormatter`
Форматирует типы занятий.
- `IRoomNameFormatter RoomNameFormatter`
Форматирует названия аудиторий.
- `ILocativeTransformer LocativeTransformer`
Преобразует локации (например, названия аудиторий и корпусов).
- `ILessonsProvider LessonsProvider`
Общий провайдер уроков.
- `IScheduleProvider ScheduleProvider`
Провайдер расписаний, объединяющий данные о днях и неделях.
- `ILessonDataFormatter LessonDataFormatter`
Форматирует данные уроков (тип, имя, место).
- `ILessonToStringFormatter LessonToStringFormatter`
Форматирует отдельный урок в строковое представление.
- `IMultiLessonToStringFormatter MultiLessonToStringFormatter`
Форматирует несколько уроков в человекочитаемую строку.