Пакет @morphcluster/carabi

Обзор

carabi — это клиентская библиотека для работы с **информационными объектами** (документами) в экосистеме CSP. Она предоставляет высокоуровневый API для взаимодействия с сервисами управления документами, а также вспомогательные утилиты для кэширования, работы со структурой и данными.

Библиотека построена поверх **NATS-интеграции** и использует `ServiceNats` для прозрачного вызова удалённых сервисов.

Клиенты

Клиенты наследуются от `ServiceNats` и предоставляют методы для вызова удалённых сервисов через NATS. Все методы принимают параметры, объект `workspace` (устаревший, всегда `null`) и логгер.

1. Documents

Назначение: Низкоуровневая работа с документами (чтение/запись значений, информация, проверки).

Методы:

Метод	Параметры	Описание
`getInfo`	`{ documentId }`	Получить основную информацию (тип, статус, дата, создатель)
`getDescr`	`{ documentId }`	Получить описание документа (по правилам именования)
`getValues`	`{ dockind, documentId, propNames }`	Получить значения указанных свойств
`setValues`	`{ dockind, documentId, values }`	Установить значения свойств
`dublicate`	`{ dockind, documentId }`	Создать копию документа
`insertRefs`	`{ dockind, documentId, propName, values }`	Добавить ссылки в множественное поле
`deleteRefs`	`{ dockind, documentId, propName, values }`	Удалить ссылки
`checkNull`	`{ dockind, documentId }`	Проверить обязательные поля

Метод	Параметры	Описание
`listIdByProps`	`{ dockind, values, count? }`	Найти ID документов по свойствам (AND)
`listPropsByProps`	`{ dockind, props, filter?, count? }`	Выбрать документы с указанными полями
`listByUniqueFields`	`{ dockind, values }`	Найти по уникальным полям
`getUniqueCollisions`	`{ dockind, documentId }`	Найти конфликты уникальности

---

2. DocumentLists

Назначение: Расширенный поиск документов с фильтрацией по статусам.

Методы:

Метод	Параметры	Описание
`listIdByProps`	`{ dockind, values, statuses?, notStatuses?, count? }`	Найти ID с фильтром по статусам
`listPropsByProps`	`{ dockind, props, filter?, statuses?, notStatuses?, count? }`	Вывести поля с фильтрацией
`listByUniqueFields`	`{ dockind, values }`	Поиск по уникальным полям
`getUniqueCollisions`	`{ dockind, documentId }`	Проверить коллизии

---

3. DockindStructure

Назначение: Управление метаданными типа документа (статусы, свойства, функции, действия, права).

Методы:

Метод	Параметры	Описание
`getInfo`	`{ docKind }`	Получить базовую информацию
`setMain`	`{ oldDocKind, newDocKind, description, parentId }`	Обновить описание и родителя
`getStatuses`	`{ docKind }`	Получить список статусов
`setStatuses`	`{ docKind, deleteIds, statuses, newOrder }`	Обновить статусы
`getProps`	`{ docKind }`	Получить свойства
`setProps`	`{ docKind, props, deleteIds }`	Обновить свойства
`getFunctions`	`{ docKind }`	Получить функции (DB/CSP)
`setFunctions`	`{ docKind, funcs, deleteIds }`	Обновить функции

Метод	Параметры	Описание
`setActions`	`{ docKind, deleteIds, actions }`	Обновить действия
`setNames`	`{ docKind, names }`	Обновить правила именования
`setPermissions`	`{ docKind, roles, deleteRoleIds }`	Обновить права доступа
`getIdByName`	`{ docKind }`	Получить ID по имени
`getNameById`	`{ docKindId }`	Получить имя по ID
`updateDockindNames`	`{}`	Отложенное обновление всех имён

Событие: `onChanged` — генерируется при любом изменении структуры.

---

4. DockindImporter

Назначение: Импорт/экспорт метаданных в форматах XML и JSON.

Методы:

Метод	Параметры	Описание
`importXml`	`{ root, vocabs, types }`	Импорт структуры из XML
`exportXml`	`{ dockind }`	Экспорт структуры в XML

---

5. DocDataImporter

Назначение: Импорт/экспорт данных документов (содержимого).

Методы:

Метод	Параметры	Описание
`importXml`	`{ XML }`	Импорт данных из XML
`exportXml`	`{ columnsXML, filterXML }`	Экспорт данных в XML

Классы - Хелперы

DocumentsHelper

Назначение: Высокоуровневый фасад для работы с документами. Предоставляет объектно-ориентированный API, скрывая детали низкоуровневых вызовов.

Зависимости: `QueriesHelper`, `DockindCache`, `Documents`, `DocumentLists`.

Методы:

Метод	Описание
`createDocument(log, session, docKindName, trxId?)`	Создать новый документ. Возвращает `Document`.
`loadDocument(log, session, docKindName, documentId, trxId?)`	Загрузить существующий документ. Возвращает `Document`.
`loadDocList(log, session, docKindName, options)`	Загрузить список документов по фильтру. Возвращает массив `Document`.
`getDocKindByDocumentId(log, documentId)`	Получить тип документа по ID документа.

Параметры `options` для `loadDocList`:

- `props: string[]` — имена свойств для вывода
- `filter: object` — фильтр по значениям
- `statuses: string[]` — ограничение по статусам
- `notStatuses: string[]` — исключение статусов
- `count: number` — максимальное количество (по умолчанию 100000)
- `trxId: number` — идентификатор транзакции (опционально)

Пример:

```
const helper = new DocumentsHelper(host);
// ... зависимости подгружаются автоматически
const doc = await helper.createDocument(log, session, 'INVOICE');
await doc.setValues({ amount: 1000 }, true); // autocommit
```

DockindCache

Назначение: Кэширование типов документов (`DocumentKind`) с автоматической загрузкой и инвалидацией.

Особенности:

- Использует `MemoryCacheAsync` с TTL 10 часов, максимум 100 типов.
- Предоставляет методы `getById(log, docKindId)` и `get(log, docKindName)`.
- Автоматически инвалидируется при изменении структуры через событие `onChanged`. Для принудительной очистки вызовите `clearCache()`.

Внутренние PromiseSingleton:

- `vocabNames` — список справочников
- `dockindNames` — список типов документов

Пример:

```
const cache = new DockindCache(host);
const dockind = await cache.get(log, 'INVOICE');
// dockind — это DocumentKind
```

DocumentKind

Назначение: Модель типа документа, загружаемая из БД. Содержит всю структуру: свойства, статусы, привязки.

Свойства:

Свойство	Тип	Описание
`id`	`number`	ID типа
`name`	`string`	Имя типа
`description`	`string`	Описание
`tableName`	`string`	Имя таблицы в БД
`properties`	`DocProperty[]`	Список свойств
`propEvents`	`DocEventProperty[]`	Привязки свойств к статусам (read/write/not null)
`statuses`	`Array<{id, name, descr}>`	Список статусов
`loaded`	`boolean`	Загружен ли объект

Методы:

Метод	Описание
`load(log)`	Загрузить структуру из БД
`getTableName()`	Получить имя таблицы
`getPropFieldName(prop)`	Получить имя поля для свойства
`getUniqueProps()`	Получить уникальные свойства

DocProperty:

- `id`, `name`, `descr` — идентификатор, имя, описание
- `propKind` — тип свойства (1 — строка, 9 — ссылка, 10 — справочник и т.д.)
- `formatBase` — базовый тип (text, numeric, timestamp, ref, int8)
- `formatLogic` — логический тип (text, numeric, date, ref, vocab)
- `multi` — множественное
- `unique` — уникальное
- `refDocKinds` — для ссылок: список допустимых типов
- `fieldName` — имя поля для синхронизации таблиц

Document

Назначение: Обёртка над документом с кэшированием значений и отслеживанием изменений.

Конструктор:

```
new Document(log, session, docKind, { Documents, QueriesHelper, trxId })
```

Свойства:

- `id` — ID документа
- `docKind` — объект `DocumentKind`
- `valuesCache` — кэш значений свойств
- `valuesChanged` — Set имён изменённых свойств

Методы:

Метод	Описание
`getValues(propNames)`	Получить значения нескольких свойств (с кэшированием)
`getValue(propName)`	Получить значение одного свойства
`setValues(values, autocommit)`	Установить значения (с отслеживанием изменений)
`commitValues()`	Сохранить накопленные изменения в БД
`insertRefs(propName, values)`	Добавить ссылки (без кэширования)
`setStatus(newStatusName)`	Сменить статус документа
`loadFromJson(data)`	Загрузить документ из JSON (полученного из `listPropsByProps`)

Важно: `setValues` не вызывает `commitValues` автоматически, если `autocommit = false`. Это позволяет группировать изменения в рамках одной транзакции.

VocabsHelper

Назначение: Загрузка справочников по имени с кэшированием и защитой от конкурентных запросов.

Методы:

Метод	Описание
`getVocab(log, vocabName)`	Получить объект `Vocab` по имени (загружается асинхронно)

Пример:

```
const helper = new VocabsHelper(host);
const vocab = await helper.getVocab(log, 'CURRENCY');
// vocab.id — ID справочника
```

Vocab

Назначение: Модель справочника (пока только загружает ID по имени).

Свойства:

- `id` — ID справочника
- `name` — имя
- `loaded` — флаг загрузки

DocInstaller

Назначение: Автоматическая установка структуры и данных документов из файлов при старте приложения. Используется для развёртывания типов документов и миграций.

DocInstaller (главный)

Зависимости: `DockindInstaller`, `DocDataInstaller`.

Методы (запросы):

Метод	Описание
`getStatus()`	Получить статус установки
`run()`	Запустить полную установку (типы + данные)
`dockindsRun()`	Запустить только установку типов
`docDataRun()`	Запустить только установку данных
`docDataGetLast()`	Получить последнюю установленную версию данных

Конфигурация:

- `skipInstall: true` — пропустить автоматическую установку при старте.

DockindInstaller

Назначение: Установка типов документов из XML-файлов.

Путь: `./dockinds/<категория>/`

Структура каталога:

```
dockinds/
  └── my_category/
     ├── category.json          # Метаданные категории (опционально)
     ├── INVOICE_types.xml      # Структура типа INVOICE
     └── INVOICE_vocabs.xml     # Справочники для типа (опционально)
```

Формат `category.json`:

```
{
  "name": "Моя категория",
  "roles": ["ROLE_ADMIN"]
}
```

Логика:

1. Сканирует подкаталоги в `./dockinds/`.
2. Для каждой категории загружает все пары `*_types.xml` и `*_vocabs.xml`.
3. Вычисляет SHA-256 хеш содержимого.
4. Сравнивает с хешем в таблице `DOCKIND_VERSIONS`.
5. Если изменилось — вызывает `DockindImporter.importXml`.
6. После успешной установки вызывает `PKG_XML_REPL_CS.REVISION_STRUCTURE`.

DocDataInstaller

Назначение: Установка данных документов из XML-файлов (миграции).

Путь: `./doc-data/<номер_версии>/*.xml`

Логика:

1. Проверяет наличие таблицы `MIGRATIONS_DATA`.
2. Сканирует подкаталоги в `./doc-data/` с числовыми именами.
3. Для каждой версии, которая ещё не установлена или была с ошибкой:
   - Выполняет все `*.xml` файлы через `DocDataImporter.importXml`.
   - В случае успеха записывает запись со статусом `done`.
   - В случае ошибки — запись со статусом `error` и текстом ошибки.
4. Если миграция завершилась ошибкой, последующие не выполняются.

Метод `resolveError`: Позволяет вручную пометить ошибочную миграцию как выполненную (перезапуск с этой версии пропускается).

Revision #5
Created 22 June 2026 19:13:51 by Admin
Updated 26 June 2026 15:40:21 by Admin