Библиотека работы с файлами

File Library

Библиотека для работы с файлами в Quarkus микросервисах.
Обеспечивает работу с файлами(Minio, seaweed) из коробки, предлагая управление как через ендпоинты, так и через сервисы.

Содержание

- Требования
- Подключение
- Использование
- Методы
- Статусы

Модули

Модуль	Описание
`file-base`	Головная библиотека для подключения в проектах
`file-core`	Реализация основного функционала
`file-provider-local`	Модуль подключения провайдера для работы с файлами локально
`file-provider-minio`	Модуль подключения провайдера для работы с файлами minio
`file-provider-seaweedfs`	Модуль подключения провайдера для работы с файлами seaweedfs
`file-rest`	Модуль управления рест-точками для работы с файлами

Требования

- Java 21+

- Quarkus 3.33.1.1
- Maven 3.9.12+

Подключение

1. Конфигурация микросервисов

1.1. Зависимости

Добавьте зависимость в `pom.xml` вашего микросервиса:

```
<dependency>
    <groupId>com.bivgroup.lib</groupId>
    <artifactId>file-base</artifactId>
    <version>${revision}</version>
</dependency>
```

1.2. Подключение Liquibase

В ресурсах проекта (например, src/main/resources/db/) создайте основной Liquibase-файл changeLog.xml и включите в него ченджсет библиотеки для работы с файлами:

```
<?xml version="1.1" encoding="UTF-8" standalone="no"?>
<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
    xmlns:pro="http://www.liquibase.org/xml/ns/pro"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog-ext
        http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-ext.xsd
        http://www.liquibase.org/xml/ns/pro
        http://www.liquibase.org/xml/ns/pro/liquibase-pro-latest.xsd
        http://www.liquibase.org/xml/ns/dbchangelog
        http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-latest.xsd">
    <include file="lib-file/lib-file-changelog.xml"/>
</databaseChangeLog>
```

В `application.properties` включите миграцию при старте:

```
quarkus.liquibase.migrate-at-start=true
```

1.3. Конфигурация

В случае использования в платформе достаточно добавить:

```
quarkus.kubernetes-config.enabled=true
quarkus.kubernetes-config.config-maps=\
  common-filestorage-cfg
```

В случае, если сервис работает вне кластера k8s
тогда в параметрах application.properties необходимо указать свойста, указанные в файле:
business-service-example.properties

default-provider - обязательный параметр используемого провайдера (всегда только один)
enabled-providers=SEAWEEDFS, MINIO - в случае необходимости использования нескольких провайдеров указываются оба через запятую. В случае, если используется только одно хранилище, то оно указывается и в biv.platform.lib.file.default-provider, и в biv.platform.lib.file.enabled-providers)

```
biv.platform.lib.file.default-provider=MINIO
biv.platform.lib.file.enabled-providers=SEAWEEDFS, MINIO
```

biv.platform.lib.file.rest-enabled - вкл/выкл рест точки "из корбки"(по-умолчанию true)

```
biv.platform.lib.file.rest-enabled=true
```

allowed-content-types - обязательный параметр для ограничения доступных для работы расширений

```
biv.platform.lib.file.validation.allowed-content-types=pdf,png,jpeg,txt,doc
```

max-size-bytes - ограничение на размер файла (по умолчанию 104857600)

```
biv.platform.lib.file.validation.max-size-bytes=104857600
```

thumbnail.max-pages - ограничение на количество страниц превью (по умолчанию 30)

```
biv.platform.lib.file.thumbnail.max-pages=30
```

При использовании провайдера seaweed необходимо указывать конфиги подключения:
providers.seaweedfs.endpoint - урл
providers.seaweedfs.bucket - бакет
providers.seaweedfs.access-key - логин
providers.seaweedfs.secret-key - пароль
В контуре платформы дефолтные значения уже указаны и их не нужно переопределять.

```
biv.platform.lib.file.providers.seaweedfs.endpoint=https://<url>:<port>
biv.platform.lib.file.providers.seaweedfs.bucket=template
biv.platform.lib.file.providers.seaweedfs.access-key=<login>
biv.platform.lib.file.providers.seaweedfs.secret-key=<password>
```

При использовании провайдера minio необходимо указывать конфиги подключения:
providers.minio.endpoint - урл
providers.minio.bucket - бакет
providers.minio.access-key - логин
providers.minio.secret-key - пароль
В контуре платформы дефолтные значения уже указаны и их не нужно переопределять.

```
biv.platform.lib.file.providers.minio.endpoint=https://<url>:<port>
biv.platform.lib.file.providers.minio.bucket=template
biv.platform.lib.file.providers.minio.access-key=<login>
biv.platform.lib.file.providers.minio.secret-key=<password>
```

Конфиги для управление процессом удаления:
biv.platform.lib.file.cleanup.enabled - вкл/выкл физического удаления файла
biv.platform.lib.file.cleanup.retention.delete-pending-retention - срок, через который физически удалять файл, помеченный на удаление
biv.platform.lib.file.cleanup.retention.thumbnail-delete-pending-retention=PT2M
biv.platform.lib.file.cleanup.retry.max-delete-attempts - количество попыток удаления файла
biv.platform.lib.file.cleanup.retry.delete-retry-delay - срок, через который повторять попытку удаления

```
biv.platform.lib.file.cleanup.enabled=false
biv.platform.lib.file.cleanup.retention.delete-pending-retention=PT2M
biv.platform.lib.file.cleanup.retention.thumbnail-delete-pending-retention=PT2I
biv.platform.lib.file.cleanup.retry.max-delete-attempts=3
biv.platform.lib.file.cleanup.retry.delete-retry-delay=PT10M
```

Использование

1. rest-точек из коробки

Классы FileResource и FileThumbnailResource предоставляют пользователю библиотеки готовые ендпоинты для работы с файлами и превью

```
@Path("/files/v3")
@POST
@Operation(summary = "Сохранить файл")
public Response upload(@RestForm("fileUpload") FileUpload fileUpload,
                       @RestForm("entityBrief") String entityBrief,
                       @RestForm("entityId") String entityId
```

```
)
@GET
@Path("/files/v3/{uuid}/meta")
@Operation(summary = "Получить метаинформацию по файлу")
public FileDto get(@PathParam("uuid") String uuid)

@GET
@Path("/files/v3/meta/list")
@Operation(summary = "Получить метаинформацию по файлам по бизнес-сущности")
public List<FileDto> list(@QueryParam("entityBrief") String entityBrief,
                          @QueryParam("entityId") String entityId)

@GET
@Path("/files/v3/{uuid}/content")
@Operation(summary = "Скачать файл")
public Response download(@PathParam("uuid") String uuid)

@DELETE
@Path("/files/v3/{uuid}")
@Operation(summary = "Пометить файла на удаление")
public Response delete(@PathParam("uuid") String uuid)

@PATCH
@Path("/files/v3/link")
@Operation(summary = "Связать файл с сущностью")
public Response delete(@QueryParam("fileId") String fileUuid,
                       @QueryParam("entityId") String entityId)

@POST
@Path("/thumbnails/v3/from-file/{fileId}")
@Operation(summary = "Создание превью для файла по идентификатору")
public Response create(@PathParam("fileId") String fileId)

@POST
@Path("/thumbnails/v3/file-with-thumbnails")
@Operation(summary = "Сохранение файла и создание превью к нему")
public Response create(@RestForm("fileUpload") @NotNull FileUpload fileUpload,
                       @RestForm("entityBrief") @NotNull String entityBrief,
                       @RestForm("entityId") @NotNull String entityId)

@POST
@Path("/thumbnails/v3/preview")
@Operation(summary = "Создание превью файла без сохранения в БД")
public Response createPreview(@RestForm("fileUpload") @NotNull FileUpload fileU

@GET
@Path("/thumbnails/v3/list-by-file/{fileId}")
@Operation(summary = "Получение списка превью по идентификатору файла")
public List<FileThumbnailDto> list(@PathParam("fileId") @NotNull String fileId
```

Вся информация об эндпоинтах есть в swagger-ui

2.Через использование методов

Для работы с файлами через интерфейс FileFacade(upload, get, download, delete, list)
Пример использованя:

```
import com.bivgroup.platform.lib.file.core.api.FileFacade;
import com.bivgroup.platform.lib.file.core.model.FileDto;
import com.bivgroup.platform.lib.file.core.model.UploadFileRequest;
import jakarta.enterprise.context.ApplicationScoped;
import jakarta.inject.Inject;

import java.io.InputStream;

@ApplicationScoped
public class FileService {

    @Inject
    FileFacade fileFacade;

    public FileDto uploadSomething(String entityId, String entityType,String f
        UploadFileRequest request = new UploadFileRequest();
        request.setFileName(filename);
        request.setEntityId(entityId);
        request.setEntityType(entityType);
        request.setContentType(contentType);
        request.setSize(size);

        return fileFacade.upload(request,content);
    }
}
```

Для работы с превью через интерфейс FileThumbnailFacade(createByFileId, createFileWithThumbnail, createThumbnailPreview, listByFileId, getPreviewInfoByFileId)

Методы

Методы доступные для работы с файлами

Метод	Описание
`upload(UploadFileRequest request, InputStream content)`	Зазгрузка файла в хранилище

Метод	Описание
`get(String fileId)`	Получение метаинформации по файлу
`download(String fileId)`	Выгрузка файла из хранилища
`delete(String fileId)`	Пометка файла на удаление
`list(String entityType, String entityId)`	Получение списка файлов, связанных с сущностью

Методы доступные для работы с превью

Метод	Описание
`createByFileId(String fileId)`	Превью по файлу из БД
`createFileWithThumbnail(FileUpload fileUpload, String entityType, String entityId)`	Сохранение файла и превью в БД
`createThumbnailPreview(FileUpload fileUpload)`	Превью без сохранения в БД
`listByFileId(String fileId)`	Список превью у файла

Методы доступные для переопределения

Метод	Описание
`generateFilename()`	Определяет логику именования файлов
`resolve(String entityType , String entityId, boolean isThumbnail)`	Определяет логику path для файла в хранилище
`resolveForUpload(UploadFileRequest request)`	Определяет логику выбора провайдера для upload
`checkUpload(FileAccessContext context)`	Определяет логику прав на загрузку

Метод	Описание
`checkRead(FileAccessContext context)`	Определяет логику прав на чтение
`checkDelete(FileAccessContext context)`	Определяет логику прав на удаление

Статусы

Константа	Описание
`ACTIVE`	Файл успешно записан
`DELETE_PENDING`	Файл помечен на удаление
`DELETED`	Файл фактически удален
`FAILED_UPLOAD`	Ошибка загрузки файла
`FAILED_DELETE`	Ошибка удаления файла
`ARCHIVE`	Файл помечен как архивный

2026 ООО "БиАйВи". Все права защищены. | Powered by Wiki.js