Reference book concept

BIV group

1.3, : Final

Table of Contents

Предназначение

Библиотека «Справочники» предназначена для реализации справочной подсистемы в микросервисной архитектуре.

Библиотека реализована как jar. В перспективе будет реализована как extension.

Архитектура

Справочник может быть необходим:

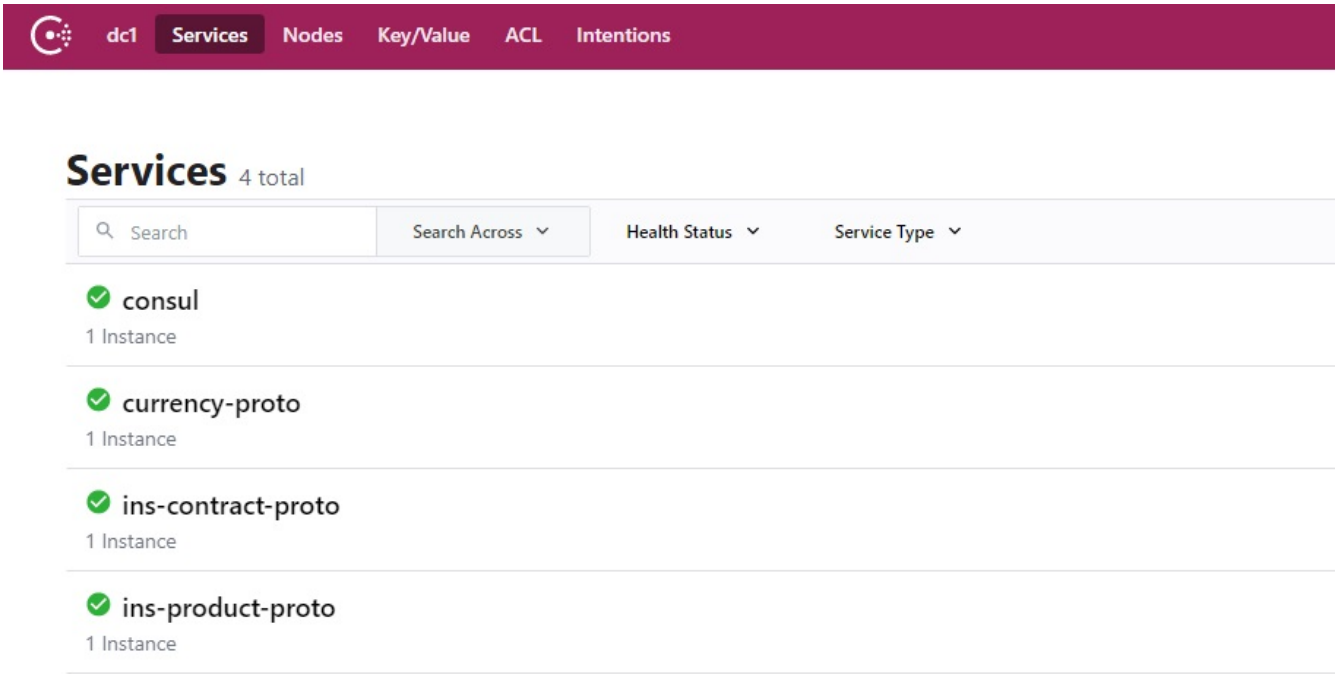
- Только в рамках одного микросервиса – это локальный справочник.
- В рамках нескольких микросервисов – общий справочник.

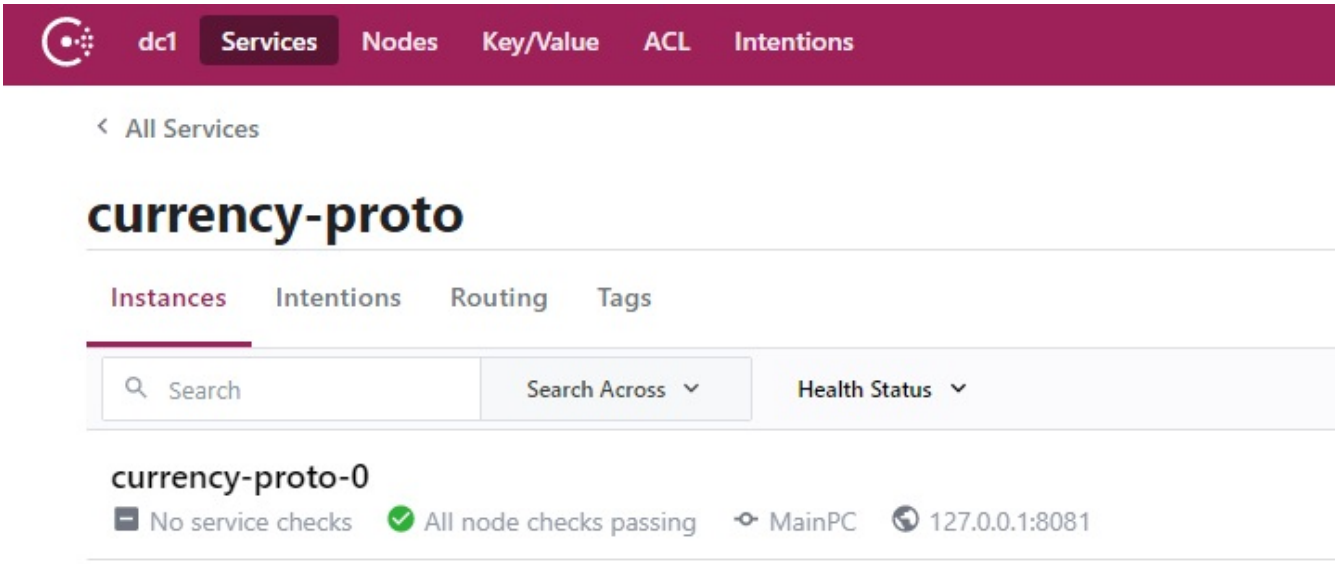
Для локальных справочников проблема межсервисного взаимодействия и разыменования ссылок не стоит. Далее рассматриваются в основном общие справочники. Из всех микросервисов, где необходим общий справочник выделяется единственный микросервис-владелец справочника. Соответственно это будет мастер-справочник. Остальные микросервисы, где необходим это справочник, будут микросервисами-потребителями справочника.

Справочник имеет алиас (т.е. псевдоним), который однозначно идентифицирует справочник в рамках всей системы (а не одного микросервиса). Псевдоним используется при описании конфигурации и в коде.

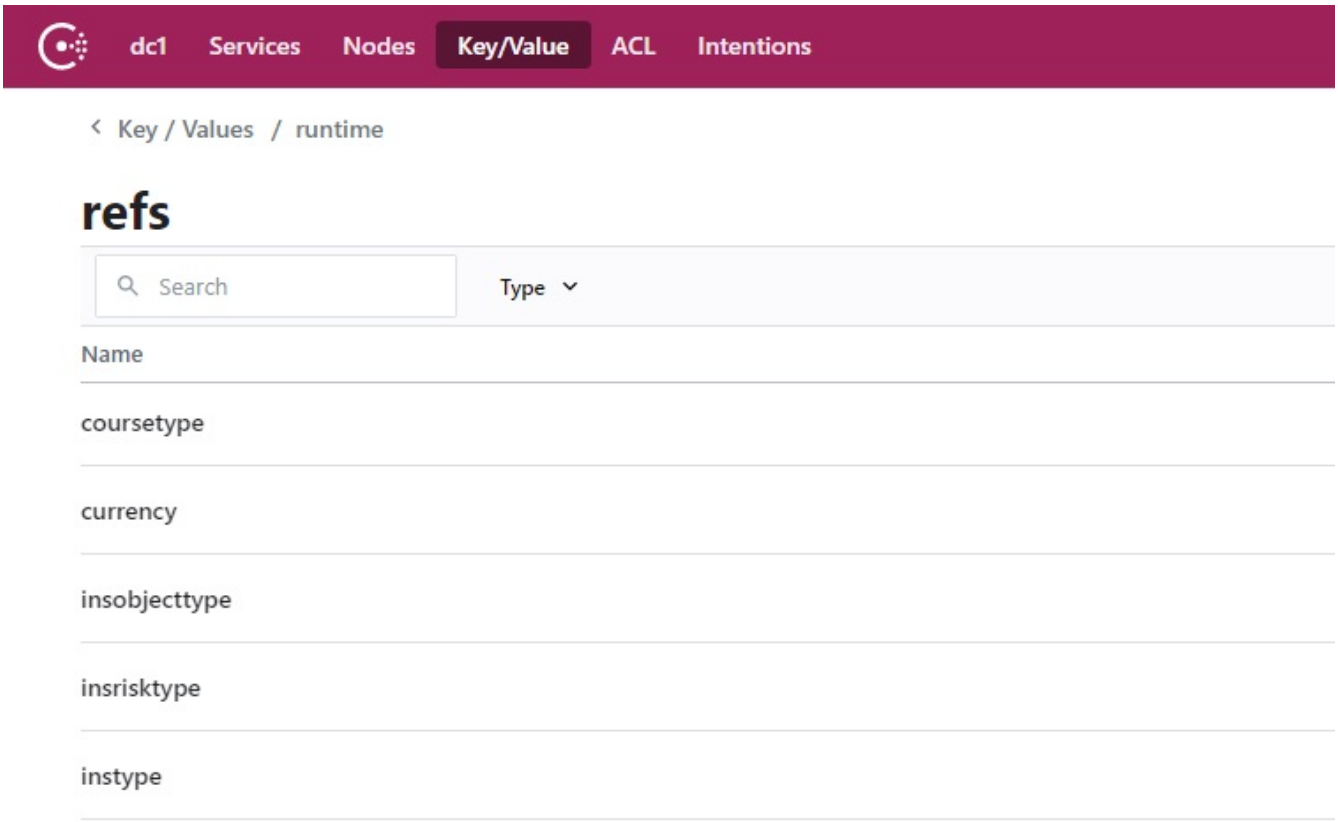
Использование service discovery для обнаружения адреса мастер-справочника

При старте микросервис регистрируется в service discovery (Consul).

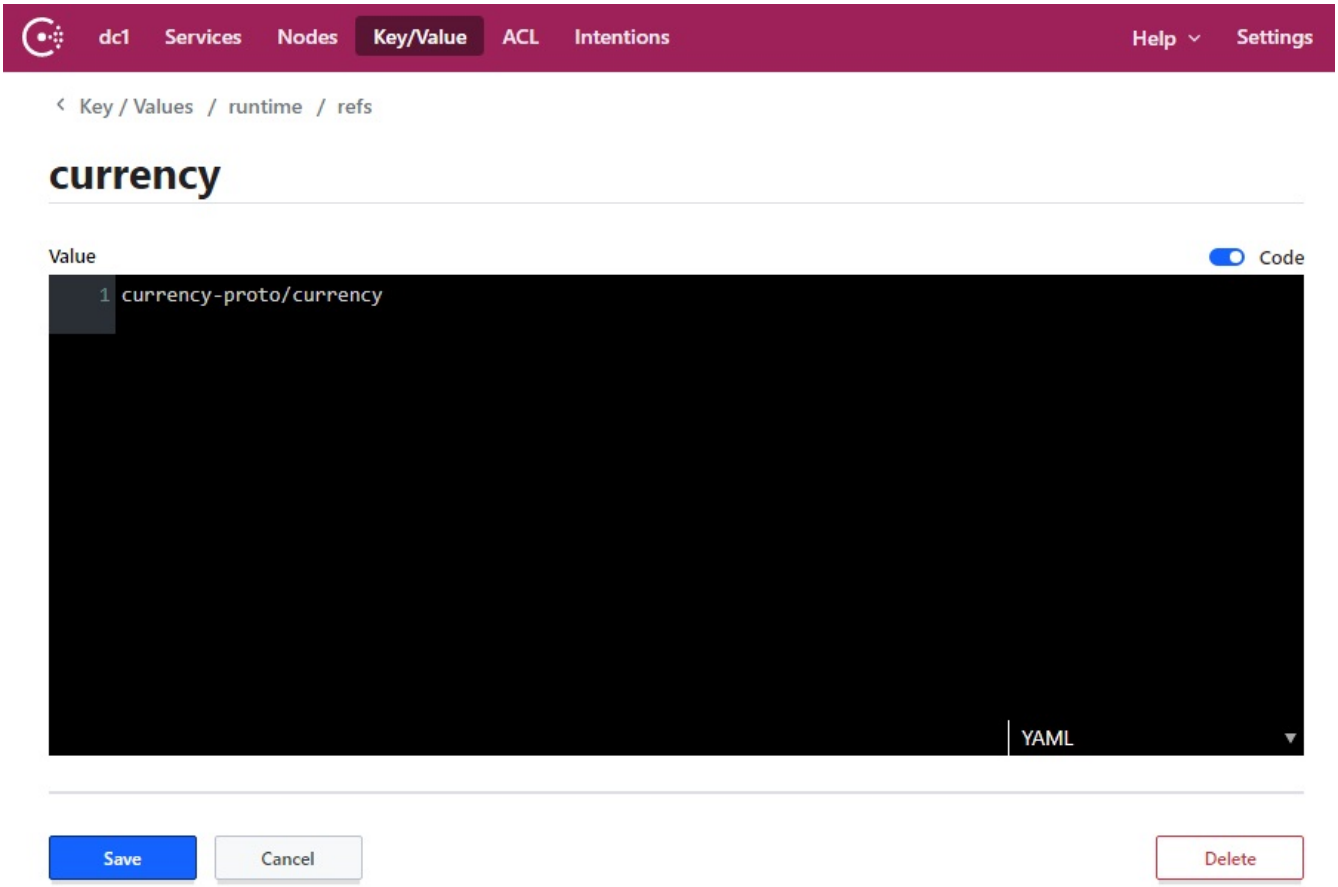




Также при старте микросервис регистрирует в service discovery (Consul) алиасы всех своих мастер-справочников. Регистрация осуществляется в Key/Value хранилище по пути *runtime* / *refs*:



В key указывается алиас справочника и путь. В value указывается имя микросервиса-владельца справочника.



Таким образом, если требуется узнать адрес микросервиса-владельца конкретного справочника, необходимо:

1. Обратиться в хранилище Key/Value [runtime/refs/алиас_справочника] consul и считать имя микросервиса-владельца и путь.

   В примере имя микросервиса-владельца=[*currency-proto*] и путь=[*/currency*]

2. обратиться в реестр сервисов и по имени сервиса узнать реальный адрес запущенного инстанса (или нескольких инстансов)

   В примере *127.0.0.1:8081*

   В результате методы справочника доступны по адресу: *http://127.0.0.1:8081/currency*

Типы доступа к мастер-справочнику

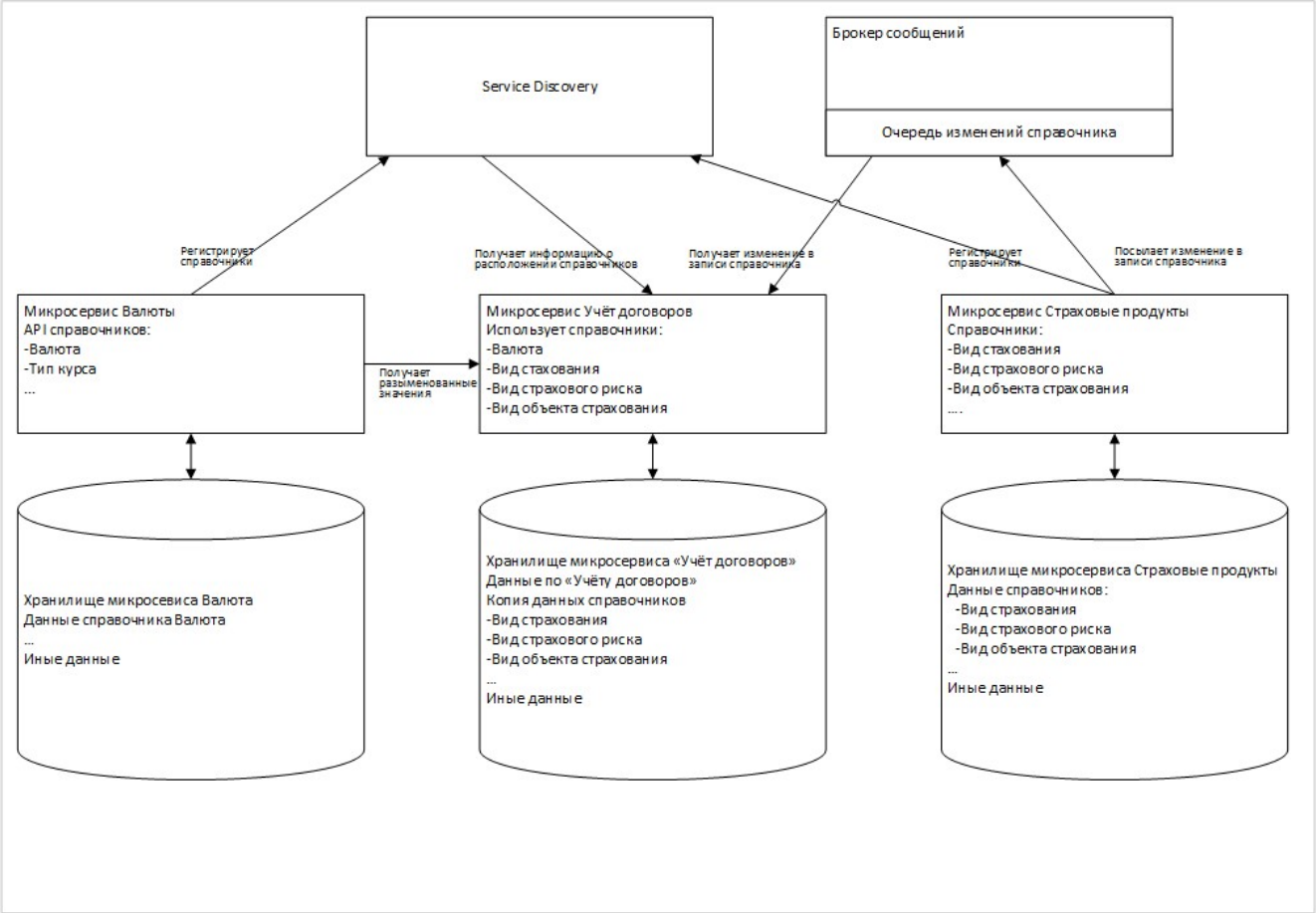
Доступ к данным мастер-справочника возможен двумя способами:

1. через API путём вызова метода через http;
2. через репликацию данных справочника из хранилища микросервиса-владельца в хранилища микросервисов-потребителей.

Репликация данных справочника

В состав системы введён Брокер сообщений (Kafka). Брокер сообщений имеет набор очередь для уведомления об изменениях данных в мастер-справочниках. При изменении данных в справочнике микросервис-владелец справочника ставит в очередь сообщение об изменениях (операция и данные). Микросервис-потребитель справочника подписывается на очередь (каждый со своим group id), забирает изменения и вносит их в копию справочника в своей БД. Иное обновление данных в «подчинённом» справочнике запрещено. Микросервис-владелец справочника имеет API для выгрузки всех данных справочника, чтобы инициализировать «подчинённый» справочник.

Пример межсервисного взаимодействия



"Валюта", "Страховые продукты", "Учёт договоров страхования".
Взаимодействие с микросервисом Валюта идёт по принципу API Composition.
Взаимодействие с микросервисом Страховые продукты идёт по принципу CQRS.

Типы справочников

Справочники делятся на:

- простые (типовые) - данные справочники имеют одинаковый атрибутный состав: Наименование, Сокращение, Код
- нетиповые - данные справочники имеют произвольный атрибутный состав

Простые справочники

Для ведения простого справочника используется общая (для всех простых справочников) таблица ref_simpleref с составным первичным ключом.

Состав первичного ключа:

- числовой идентификатор - формируется с помощью последовательности отдельной simpleref_sequence
- дискриминатор, разделяющий записи разных справочников в рамках одной таблицы.

Комбинация ИД + дискриминатор обеспечивает одинаковую ид записей вне зависимости от микросервиса.

Базовая функциональность простых справочников реализована в следующих классах:

- BaseRefResource - базовый класс, от которого необходимо отнаследоваться, чтобы получить http-ресурс с методами по работе со справочниками (и простыми и нетиповыми).
- SimpleRefService и SimpleRefServiceImpl - интерфейс и его реализация по взаимодействию с базой данных.
- SimpleRef - базовый класс сущности простого справочника. От него необходимо отнаследоваться при создании реального справочника.
- SimpleRefPK - класс, описывающий первичный ключ сущности SimpleRef.
- SimpleRefDTO - data transfer object для простого справочника. Наследование не требуется.

Более подробно разработка описана в разделах:

- Простой справочник в микросервисе-владельце справочника
- Простой справочник в микросервисе-потребителе справочника с репликацией

Нетиповые справочники

Для ведения нетипового справочника необходимо создать таблицу в базе данных.

Необходимо создать сущность.

Необходимо создать data transfer object унаследованный от BaseDTO.

Базовый сервис по работе со справочником реализован в библиотеке. При корректном описании справочника базовый сервис будет выполнять основные действия. Если базовой функциональности не хватает, то можно реализовать собственный сервис.

Базовый ресурс по работе со справочником также реализован в библиотеке. Его методы:

```
@POST
@Path("/{refName}/getById")
public BaseResponse<BaseDTO> getById(@PathParam("refName") String refName, IdDTO
```

```
request)...

@POST
@Path("/{refName}/listAll")
public BaseResponse<List<BaseDTO>> listAll(@PathParam("refName") String refName)...

@POST
@Path("/{refName}/orderedListAll")
public BaseResponse<List<BaseDTO>> orderedListAll(@PathParam("refName") String
refName, OrderByDTO request)...

@POST
@Path("/{refName}/findAll")
public BaseResponse<List<BaseDTO>> findAll(@PathParam("refName") String refName,
QueryCriteriaDTO request)...

@POST
@Path("/refbooks/listAll")
public BaseResponse<Map<String, List<BaseDTO>>> multiListAll(MultiAliasDTO request)...

@POST
@Path("/refbooks/orderedListAll")
public BaseResponse<Map<String, List<BaseDTO>>> multiOrderedListAll(MultiOrderByDTO
request)...

@POST
@Path("/refbooks/findAll")
public BaseResponse<Map<String, List<BaseDTO>>> multiFindAll(MultiQueryCriteriaDTO
request)...

@POST
@Path("/{refName}/add")
@Transactional
public BaseResponse<BaseDTO> add(@PathParam("refName") String refName, String
requestStr) throws ServiceException, ReflectiveOperationException,
JsonProcessingException...

@POST
@Path("/{refName}/update")
@Transactional
public BaseResponse<BaseDTO> update(@PathParam("refName") String refName, String
requestStr) throws ServiceException, ReflectiveOperationException,
JsonProcessingException...

@POST
@Path("/{refName}/delete")
@Transactional
public BaseResponse<Long> delete(@PathParam("refName") String refName, IdDTO request)
...
```

Более подробно разработка описана в разделах:

- Нетиповой справочник в микросервисе-владельце справочника
- Нетиповой справочник в микросервисе-потребителе справочника с доступом по API

Конфигурирование

Конфигурирование возможно через:

- Только файл application.properties
- Файл application.properties и таблицу refbookconfig в базе данных

Параметры в application.properties, необходимые при любом способе конфигурации

Параметры общего назначения:

```
quarkus.application.name=ins-contract
```

Параметры для взаимодействия с consul:

```
# consul
quarkus.consul-config.enabled=true
quarkus.consul-config.agent.host-port=localhost:8500
# quarkus.consul-config.properties-value-keys=config/${quarkus.application.name}
```

Параметр *quarkus.consul-config.properties-value-keys* требуется в случае, если в consul будут храниться какие-либо значения из application.properties.

Параметры подключения к базе данных:

```
# datasource
quarkus.datasource.db-kind=postgresql
quarkus.datasource.username=user_name
quarkus.datasource.password=1
quarkus.datasource.jdbc.url=jdbc:postgresql://host_name:5432/database_name?currentSche
ma=schema_name
quarkus.datasource.jdbc.min-size=1
quarkus.datasource.jdbc.max-size=5
```

Размер пула соединений (jdbc.min-size и jdbc.max-size) указан для простой тестовой конфигурации.

Параметры hibernate:

```
# hibernate
quarkus.hibernate-orm.database.generation=none
```

Генерация элементов базы данных отключена.

Параметры Kafka:

```
# kafka messages
kafka.bootstrap.servers=host_name:9092
# outgoing
mp.messaging.outgoing.out-ref-change.connector=smallrye-kafka
mp.messaging.outgoing.out-ref-change.topic=ref-change
mp.messaging.outgoing.out-ref-change.merge=true
mp.messaging.outgoing.out-ref-
change.value.serializer=org.apache.kafka.common.serialization.StringSerializer
# incoming
mp.messaging.incoming.in-ref-change.connector=smallrye-kafka
mp.messaging.incoming.in-ref-change.topic=ref-change
# use application name as groupid
mp.messaging.incoming.in-ref-change.group.id=${quarkus.application.name}
mp.messaging.incoming.in-ref-
change.value.deserializer=org.apache.kafka.common.serialization.StringDeserializer
mp.messaging.incoming.in-ref-change.auto.offset.reset=earliest
```

kafka.bootstrap.servers - подключить брокер сообщений kafka *mp.messaging.outgoing.out-ref-change.** - для исходящих сообщений об изменениях в справочниках (в этом случае микросервис является мастер-сервисом для справочника) *mp.messaging.incoming.in-ref-change.** - для входящих сообщений об изменениях в справочниках (в этом случае микросервис является потребителем справочника)

Параметр для запуска планировщика задач quartz:

```
# quartz for initialize replication job
quarkus.quartz.force-start=true
```

Общие параметры конфигурации справочников:

```
# use database for ref-book configuration
com.bivgroup.ref.config.database=false
# ref-book defenition
com.bivgroup.ref.base.url=inscontract
```

com.bivgroup.ref.config.database - конфигурировать справочники через таблицу в базе данных (true) или через файл application.properties (false).
com.bivgroup.ref.base.url - базовый путь в URL для ресурсов со справочниками.
Например, если значение параметра равно "insproduct", то путь к методу getById справочника insobjecttype будет таким:
http://hostname:8080/insproduct/insobjecttype/getById

Основные термины при конфигурировании конкретного справочника

Алиас - сокращение, псевдоним справочника. Пишется в одно слово английскими буквами.

***direction**=[export|import]* - "направление" для справочника:
export - микросервис является мастер-сервисом для справочника и данные справочника экспортируются другим микросервисам.
import - микросервис является потребителем справочника и данные справочника импортируются.

***access**=[api|replicate]* - способ доступа к данным справочника:
api - через межсервисный вызов API.
replicate - через репликацию данных.

***simple**=[false|true]* - является ли справочник простым.

name - наименование справочника

discriminator - поле-дискриминатор для разделения записей простых справочников в рамках одной таблицы.
Применимо только для простых справочников.
Как правило алиас и дискриминатор совпадают.

entityClassName - имя класса сущности справочника.
Необходимо заполнять, если справочник хранится в базе данных и не является простым справочником.
Для простых справочников или при импорте по API параметр заполнять не надо.

dtoClassName - имя класса Data Transfer Object (pojo)
Необходимо заполнять всегда.

Конфигурация справочников через файл application.properties

```
com.bivgroup.ref.aliases=Справочник1;Справочник2;Справочник3;...;СправочникN
```

Перечислить алиасы ВСЕХ справочники, используемые в микросервисе вне зависимости от их направления и способа доступа. Далее эти алиасы используются при описании каждого справочника по отдельности.

```
com.bivgroup.ref.АЛИАС.direction=[export|import]
com.bivgroup.ref.АЛИАС.access=[api|replicate]
com.bivgroup.ref.АЛИАС.simple=[false|true]
com.bivgroup.ref.АЛИАС.name=Наименование справочника
com.bivgroup.ref.АЛИАС.discriminator=INSTYPE
com.bivgroup.ref.АЛИАС.entity.class.name=имя класса сущности справочника
com.bivgroup.ref.АЛИАС.dto.class.name=имя класса Data Transfer Object
```

Пример описания справочника Валюта, который располагается в микросервисе Валюта, и доступ к которому осуществляется через API:

```
com.bivgroup.ref.currency.direction=export
com.bivgroup.ref.currency.access=api
com.bivgroup.ref.currency.simple=false
com.bivgroup.ref.currency.name=Валюта
com.bivgroup.ref.currency.entity.class.name=com.bivgroup.ms.currency.entity.Currency
com.bivgroup.ref.currency.dto.class.name=com.bivgroup.ms.currency.dto.CurrencyDTO
```

discriminator не указан, т.к. это НЕ простой справочник, а определяемый полностью (у Валюты есть атрибут Точность).

Пример описания справочника Валюта, который импортируется в микросервис Учёт договоров страхования, и доступ к которому осуществляется через API:

```
com.bivgroup.ref.currency.direction=import
com.bivgroup.ref.currency.access=api
com.bivgroup.ref.currency.simple=false
com.bivgroup.ref.currency.name=Валюта
com.bivgroup.ref.currency.dto.class.name=com.bivgroup.ms.inscontract.dto.CurrencyDTO
```

entity.class.name не указан, т.к. в этом микросервисе справочник не хранится в базе данных, а доступ к значениям осуществляется при помощи межсервисных вызовов.
discriminator не указан, т.к. т.к. это НЕ простой справочник.

Пример описания справочника Валюта, который импортируется в микросервис Учёт договоров страхования, и доступ к которому осуществляется через репликацию:

```
com.bivgroup.ref.currency.direction=import
com.bivgroup.ref.currency.access=replicate
com.bivgroup.ref.currency.simple=false
com.bivgroup.ref.currency.name=Валюта
com.bivgroup.ref.currency.entity.class.name=com.bivgroup.ms.currency.entity.Currency
com.bivgroup.ref.currency.dto.class.name=com.bivgroup.ms.inscontract.dto.CurrencyDTO
```

discriminator не указан, т.к. т.к. это НЕ простой справочник.

Пример описания простого справочника Вид страхования в мастер-сервисе Страховые продукты, данные которого будут реплицироваться:

```
com.bivgroup.ref.instype.direction=export
com.bivgroup.ref.instype.access=replicate
com.bivgroup.ref.instype.simple=true
com.bivgroup.ref.instype.name=Вид страхования
com.bivgroup.ref.instype.discriminator=INSTYPE
com.bivgroup.ref.instype.entity.class.name=com.bivgroup.ms.insproduct.entity.InsType
com.bivgroup.ref.instype.dto.class.name=com.bivgroup.lib.refbook.dto.SimpleRefDTO
```

Пример описания простого справочника Вид страхования в микросервисе Учёт договоров страхования:

```
com.bivgroup.ref.instype.direction=import
com.bivgroup.ref.instype.access=replicate
com.bivgroup.ref.instype.simple=true
com.bivgroup.ref.instype.name=Вид страхования
com.bivgroup.ref.instype.discriminator=INSTYPE
com.bivgroup.ref.instype.entity.class.name=com.bivgroup.ms.inscontract.entity.InsType
com.bivgroup.ref.instype.dto.class.name=com.bivgroup.lib.refbook.dto.SimpleRefDTO
```

Конфигурация справочников через таблицу refbookconfig в базе данных

Для хранения конфигурации справочников используется таблица refbooconfig.
Поля таблицы:

```
alias
direction
access
initdate
simple
name
discriminator
entityclassname
dtoclassname
```

В поле initdate хранится дата и время инициализации данных импортируемого справочника.
Чтобы отключить инициализацию копии справочника, необходимо прописать в поле любую дату (значение не null).

Конфигурирование Consul

<under construction>

Конфигурирование Kafka

Для репликации данных необходимо создать топик "ref-change".

```
bin/kafka-topics.sh --create --topic "ref-change" --bootstrap-server hostname:9092
```

Разработка справочников

Для работы с библиотекой необходимо подключить зависимость в pom.xml:

```
<dependency>
    <groupId>com.bivgroup.lib</groupId>
    <artifactId>refbook</artifactId>
    <version>1.2-FINAL</version>
</dependency>
```

Нетиповой справочник в микросервисе-владельце справочника

Реализация нетипового справочника (с произвольными атрибутами) на примере сущности "Валюта" в микросервисе "Валюта".

1. Создать таблицу в базе данных.

```
CREATE TABLE schema_name.cur_currency (
    id int8 NOT NULL,
    brief varchar(10) NOT NULL,
    code varchar(10) NULL,
    "name" varchar(50) NOT NULL,
    prec int4 NULL,
    CONSTRAINT cur_currency_pkey PRIMARY KEY (id)
);
CREATE UNIQUE INDEX cur_currency_brief_idx ON schema_name.cur_currency USING btree
(brief);
```

2. Создать класс сущности.

```
@Entity
@Table(name = "cur_currency")
@Getter
@Setter
public class Currency {

    @Id
    @GeneratedValue
    private Long id;

    @Column(name = "name")
    private String name;

    @Column(name = "brief")
    private String brief;

    @Column(name = "code")
    private String code;

    @Column(name = "prec")
    private Integer prec;
}
```

Для упрощения и автоматической генерации getter`ов и setter`ов используется библиотека Project Lombok.

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.18</version>
</dependency>
```

3. Создать класс dto.

```
@JsonPropertyOrder({
    "id",
    "name",
    "brief",
    "code",
    "prec"
})
@Getter
@Setter
public class CurrencyDTO extends BaseDTO {

    @Schema(description = "Наименование")
    @JsonProperty("name")
    private String name;

    @Schema(description = "Сокращение")
    @JsonProperty("brief")
    private String brief;

    @Schema(description = "Код")
    @JsonProperty("code")
    private String code;

    @Schema(description = "Точность")
    @JsonProperty("prec")
    private Integer prec;
}
```

4. Создать конфигурацию справочника (в application.properties или в таблице refbookconfig).

   См. раздел "Конфигурирование".

   Например:

```
com.bivgroup.ref.currency.direction=export
com.bivgroup.ref.currency.access=api
com.bivgroup.ref.currency.simple=false
com.bivgroup.ref.currency.name=Валюта
com.bivgroup.ref.currency.entity.class.name=com.bivgroup.ms.currency.entity.Currenc
y
com.bivgroup.ref.currency.dto.class.name=com.bivgroup.ms.currency.dto.CurrencyDTO
```

5. В библиотеке реализован готовый сервис, работающий с нетиповыми справочниками по их описанию. Данный сервис реализует интерфейс:

```
public interface BaseRefService {

    public BaseDTO deserializeDto(String alias, String request) throws
ServiceException, ReflectiveOperationException, JsonProcessingException;

    public BaseDTO getById(String alias, IdDTO request) throws ServiceException,
ReflectiveOperationException;

    public List<BaseDTO> listAll(String alias) throws ServiceException,
ReflectiveOperationException;

    public List<BaseDTO> orderedListAll(String alias, OrderByDTO request) throws
ServiceException, ReflectiveOperationException;

    public List<BaseDTO> findAll(String alias, QueryCriteriaDTO request) throws
ServiceException, ReflectiveOperationException;

    public BaseDTO add(String alias, BaseDTO request) throws ServiceException,
ReflectiveOperationException;

    public BaseDTO update(String alias, BaseDTO request) throws ServiceException,
ReflectiveOperationException;

    public Long delete(String alias, IdDTO request) throws ServiceException,
ReflectiveOperationException;
}
```

Если возможностей данного сервиса не хватает, то можно реализовать собственный сервис по работе с базой данных, например:

```
@ApplicationScoped
public class CurrencyService implements PanacheRepository<Currency> {

    public CurrencyDTO getById(IdDTO request) throws ServiceException,
ReflectiveOperationException {
        Long id = request.getId();
        Currency currency = findById(id);
        if (currency == null) {
            throw new ServiceException(String.format("Не найдена валюта с id=%d",
id));
        }
        CurrencyDTO result = new CurrencyDTO();
        ObjectFieldMapper.copyFields(currency, result);
        return result;
    }

    public List<CurrencyDTO> listAll() throws ServiceException,
ReflectiveOperationException
        ...
    }

    public CurrencyDTO add(CurrencyDTO request) throws ServiceException,
ReflectiveOperationException {
        ...
    }

    public CurrencyDTO update(CurrencyDTO request) throws ServiceException,
ReflectiveOperationException {
        ...
    }

    public Long delete(IdDTO request) throws ServiceException,
ReflectiveOperationException {
        ...
    }
}
```

6. Базовый ресурс по работе со справочником также реализован в библиотеке. Достаточно отнаследоваться от *BaseRefResource*.

```
@Path("/currency")
@Tag(name = "Методы сервиса Валюты")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public class CurrencyResource<CurrencyDTO> extends BaseRefResource {

}
```

Если возможностей не хватает, то можно реализовать собственный ресурс, например:

```
@Path("/currency/currency")
@Tag(name = "Методы сервиса Валюты")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public class CurrencyResource {

    @Inject
    BaseResource baseResource;

    @Inject
    CurrencyService currencyService;

    @POST
    @Path("/getById")
    public BaseResponse<CurrencyDTO> getById(IdDTO request) throws
ServiceException, ReflectiveOperationException {
        CurrencyDTO reuslt = currencyService.getById(request);
        BaseResponse<CurrencyDTO> response = baseResource.buildSuccessResponse
(reuslt);
        return response;
    }

    @POST
    @Path("/listAll")
    public BaseResponse<List<CurrencyDTO>> listAll() throws ServiceException,
ReflectiveOperationException {
        ...
    }

    @POST
    @Path("/add")
    @Transactional
    public BaseResponse<CurrencyDTO> add(CurrencyDTO request) throws
ServiceException, ReflectiveOperationException {
        ...
    }

    @POST
    @Path("/update")
    @Transactional
    public BaseResponse<CurrencyDTO> update(CurrencyDTO request) throws
ServiceException, ReflectiveOperationException {
        ...
    }

    @POST
    @Path("/delete")
    @Transactional
```

```
    public BaseResponse<Long> delete(IdDTO request) throws ServiceException,
ReflectiveOperationException {
        ...
    }
}
```

7. Если справочники взаимосвязаны, например, "Валюта" ссылается на "Тип валюты", а "Курс" ссылается на "Тип курса" и "Валюту", то можно рассмотреть 2 случая.

   a. Нетиповой справочник ссылается на нетиповой справочник, например "Курс" ссылается на "Тип курса" и "Валюту" (приводимую, приведённую). Тогда класс "Курс" выглядит так:

```
@Entity
@Table(name = "cur_course")
@Getter
@Setter
public class Course {

    @Id
    @GeneratedValue
    @Column(name = "id")
    private Long id;
    ...

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(foreignKey = @ForeignKey(ConstraintMode.NO_CONSTRAINT), name =
"course_type_id")
    private CourseType courseType;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(foreignKey = @ForeignKey(ConstraintMode.NO_CONSTRAINT), name =
"from_currency_id")
    private Currency fromCurrency;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(foreignKey = @ForeignKey(ConstraintMode.NO_CONSTRAINT), name =
"to_currency_id")
    private Currency toCurrency;

    ...
}
```

   В данном случае имеется LAZY чтение зависимых данных, и в итоге ссылочные объекты в ответе будут пустыми. Чтобы заставить читаться зависимые данные необходимо установить специальный *FetchProfile*. Делается это аннотацией на классе сущности:

```
@Entity
@Table(name = "cur_course")
@FetchProfile(
        name = com.bivgroup.lib.refbook.entity.Const.REF_BOOK_FETCH_PROFILE,
        fetchOverrides = {
            @FetchProfile.FetchOverride(entity = Course.class, association =
"courseType", mode = FetchMode.JOIN)
            @FetchProfile.FetchOverride(entity = Course.class, association =
"fromCurrency", mode = FetchMode.JOIN),
            @FetchProfile.FetchOverride(entity = Course.class, association =
"toCurrency", mode = FetchMode.JOIN)
        })
@Getter
@Setter
public class Course {
    ...
}
```

где *com.bivgroup.lib.refbook.entity.Const.REF_BOOK_FETCH_PROFILE* - это зарезервированное имя fetch-профиля.

b. Нетиповой справочник ссылается на простой справочник, например "Валюта" ссылается на "Тип валюты". Тогда класс "Валюта" должен выглядеть так:

```
@Entity
@Table(name = "cur_currency")
@FetchProfile(
        name = Const.REF_BOOK_FETCH_PROFILE,
        fetchOverrides = {
            @FetchProfile.FetchOverride(entity = Currency.class, association =
"currencyType", mode = FetchMode.JOIN)
        })
@Getter
@Setter
public class Currency {

    @Id
    @GeneratedValue
    private Long id;
    ...

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumnsOrFormulas(value = {
        @JoinColumnOrFormula(column = @JoinColumn(name = "currency_type_id",
referencedColumnName = "id")),
        @JoinColumnOrFormula(formula = @JoinFormula(referencedColumnName =
"discriminator", value = "'CURRENCYTYPE'"))
    })
    private CurrencyType currencyType;
}
```

Соединение с простым справочником описывается при помощи аннотации *JoinColumnsOrFormulas*, в которой помимо колонки с идентификатором значения справочника [*currency_type_id*] указывается дискриминатор справочника [*referencedColumnName = "discriminator", value = "'CURRENCYTYPE'"*].

Простой справочник в микросервисе-владельце справочника

Реализация простого справочника на примере сущности "Вид страхования" в микросервисе "Страховые продукты".

1. Создать таблицу в базе данных, если она ещё не создана.

```
CREATE TABLE schema_name.ref_simpleref (
    id int8 NOT NULL,
    discriminator varchar(50) NOT NULL,
    "name" varchar(255) NULL,
    brief varchar(255) NOT NULL,
    code varchar(255) NULL,
    deleted bool NOT NULL,
    CONSTRAINT ref_simpleref_pkey PRIMARY KEY (id, discriminator)
);
CREATE INDEX ref_simpleref_brief_idx ON schema_name.ref_simpleref USING btree
(discriminator, brief);
CREATE INDEX ref_simpleref_name_idx ON schema_name.ref_simpleref USING btree
(discriminator, name);
```

2. Создать класс сущности.

```
@Entity
@DiscriminatorValue("INSTYPE")
public class InsType extends SimpleRef {

}
```

3. Создать конфигурацию справочника (в application.properties или в таблице refbookconfig).

   См. раздел "Конфигурирование".

   Например:

```
com.bivgroup.ref.instype.direction=export
com.bivgroup.ref.instype.access=replicate
com.bivgroup.ref.instype.simple=true
com.bivgroup.ref.instype.name=Вид страхования
com.bivgroup.ref.instype.discriminator=INSTYPE
com.bivgroup.ref.instype.entity.class.name=com.bivgroup.ms.insproduct.entity.InsTyp
e
com.bivgroup.ref.instype.dto.class.name=com.bivgroup.lib.refbook.dto.SimpleRefDTO
```

4. Создать http-ресурс для простых справочников, если он ещё не создан.

   На микросервис необходим один универсальный ресурс для работы с простыми справочниками. В классе базовые методы реализованы на уровне *BaseRefResource*.

```
@Path("/insproduct")
@Tag(name = "Методы сервиса справочников страховых продуктов")
public class RefResource extends BaseRefResource {

}
```

Нетиповой справочник в микросервисе-потребителе справочника с доступом по API

Реализация потребления нетипового справочника (с произвольными атрибутами) на примере сущности Валюта в микросервисе "Учёт договоров страхования".

1. Создать класс dto.

```
@JsonPropertyOrder({
    "id",
    "name",
    "brief",
    "code",
    "prec"
})
@Getter
@Setter
public class CurrencyDTO extends BaseDTO {

    @Schema(description = "Наименование")
    @JsonProperty("name")
    private String name;

    @Schema(description = "Сокращение")
    @JsonProperty("brief")
    private String brief;

    @Schema(description = "Код")
    @JsonProperty("code")
    private String code;

    @Schema(description = "Точность")
    @JsonProperty("prec")
    private Integer prec;
}
```

2. Создать конфигурацию справочника (в application.properties или в таблице refbookconfig).

   См. раздел "Конфигурирование".

Например:

```
com.bivgroup.ref.currency.direction=import
com.bivgroup.ref.currency.access=api
com.bivgroup.ref.currency.simple=false
com.bivgroup.ref.currency.name=Валюта
com.bivgroup.ref.currency.dto.class.name=com.bivgroup.ms.inscontract.dto.CurrencyDT
O
```

3. Вызвать метод микросервиса "Валюта" по получению данных.

```
@ApplicationScoped
public class ...Service  {

    @Inject
    RefClient refClient;

    private void callCurrencyMethods() throws JsonProcessingException,
URISyntaxException, ServiceException {
        ...
        Long currId = 1L;
        CurrencyDTO currency = (CurrencyDTO) refClient.getById("currency", currId);
        ...
        List<CurrencyDTO> currencyList = (List<CurrencyDTO>) refClient.listAll(
"currency");
        ...
}
```

Простой справочник в микросервисе-потребителе справочника с репликацией

Реализация потребления простого справочника на примере сущности "Вид страхования".

1. Создать таблицу в базе данных, если она ещё не создана.

```
CREATE TABLE schema_name.ref_simpleref (
    id int8 NOT NULL,
    discriminator varchar(50) NOT NULL,
    "name" varchar(255) NULL,
    brief varchar(255) NOT NULL,
    code varchar(255) NULL,
    deleted bool NOT NULL,
    CONSTRAINT ref_simpleref_pkey PRIMARY KEY (id, discriminator)
);
CREATE INDEX ref_simpleref_brief_idx ON schema_name.ref_simpleref USING btree
(discriminator, brief);
CREATE INDEX ref_simpleref_name_idx ON schema_name.ref_simpleref USING btree
(discriminator, name);
```

2. Создать конфигурацию справочника (в application.properties или в таблице refbookconfig).

   См. раздел "Конфигурирование".

   Например:

```
com.bivgroup.ref.instype.direction=import
com.bivgroup.ref.instype.access=replicate
com.bivgroup.ref.instype.simple=true
com.bivgroup.ref.instype.name=Вид страхования
com.bivgroup.ref.instype.discriminator=INSTYPE
com.bivgroup.ref.instype.entity.class.name=com.bivgroup.ms.inscontract.entity.InsTy
pe
com.bivgroup.ref.instype.dto.class.name=com.bivgroup.lib.refbook.dto.SimpleRefDTO
```

3. Для того, чтобы реализовать связь на уровне сущности необходимо объявить ссылку следующим образом:

```
@Entity
@Table(name = "ict_contract")
@Getter
@Setter
public class InsContract {

    @Id
    @Column(name = "id")
    private Long id;
    ...
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumnsOrFormulas(value = {
        @JoinColumnOrFormula(column
                = @JoinColumn(name = "insrisktypeid",
                        referencedColumnName = "id")),
        @JoinColumnOrFormula(formula
                = @JoinFormula(referencedColumnName = "discriminator", value =
"'INSRISKTYPE'"))
    })
    private InsType insType;
}
```

Скрипты для базы данных

СУБД PostgreSql

Вместо schema_name необходимо подставить имя схемы (по умолчанию это public).

ref_simpleref - таблица для простых справочников

```
CREATE TABLE schema_name.ref_simpleref (
    id int8 NOT NULL,
    discriminator varchar(50) NOT NULL,
    "name" varchar(255) NULL,
    brief varchar(255) NOT NULL,
    code varchar(255) NULL,
    deleted bool NOT NULL DEFAULT false,
    CONSTRAINT ref_simpleref_pkey PRIMARY KEY (id, discriminator)
);
CREATE INDEX ref_simpleref_brief_idx ON ins_product.ref_simpleref USING btree
(discriminator, brief);
CREATE INDEX ref_simpleref_name_idx ON ins_product.ref_simpleref USING btree
(discriminator, name);
```

simpleref_sequence - последовательность для генерации id в простых справочниках

```
CREATE SEQUENCE schema_name.simpleref_sequence
    INCREMENT BY 1
    MINVALUE 1
    MAXVALUE 9223372036854775807
    START 1
    CACHE 1
    NO CYCLE;
```

refbookconfig - таблица для конфигурирования справочников

```
CREATE TABLE schema_name.refbookconfig (
    alias varchar(50) NOT NULL,
    direction varchar(10) NOT NULL,
    "access" varchar(10) NOT NULL,
    initdate timestamp(0) NULL,
    "simple" bool NOT NULL DEFAULT false,
    name varchar(100) NULL,
    discriminator varchar(50) NULL,
    entityclassname varchar(100) NULL,
    dtoclassname varchar(100) NULL,
    CONSTRAINT refbookconfig_pk PRIMARY KEY (alias)
);
```

Пример справочников

Описание проектов примера

Пример содержит 3 микросервиса в каталоге "example":

- Микросервис "Валюта", проект "currency-proto"
  порт микросервиса 8081
- Микросервис "Страховые продукты", проект "ins-contract-proto"
  порт микросервиса 8082
- Микросервис "Учёт договоров страхования", проект "ins-product-proto"
  порт микросервиса 8083

В микросервисах "Валюта" и "Страховые продукты" ведутся справочники, в микросервисе "Учёт договоров страхования" справочники используются.

Состав проектов примера

Микросервис "Валюта"

1. Тип валюты - currencytype
   Простой справочник
   Мастер-справочник
   Доступ к справочнику по API
2. Валюта - currency
   Нетиповой справочник
   Мастер-справочник
   Доступ к справочнику по API Ссылается на "Тип валюты"
3. Тип курса - coursetype
   Нетиповой справочник (для примера, по сути это простой справочник)
   Мастер-справочник
   Данные справочника реплицируются
4. Курс - course
   Нетиповой справочник (для примера, по сути это не справочник, а учётные данные)
   Мастер-справочник
   Доступ к справочнику по API Ссылается на "Валюту"

Микросервис "Страховые продукты"

1. Вид страхования - instype
   Простой справочник
   Мастер-справочник
   Доступ к справочнику по API
2. Вид объекта страхования - insobjecttype Простой справочник

Мастер-справочник
Доступ к справочнику по API

3. Вид страхового риска - insrisktype Простой справочник
Мастер-справочник
Данные справочника реплицируются

Микросервис "Учёт договоров страхования"

1. Валюта - currency
Нетиповой справочник
Потребление справочника
Доступ к справочнику по API
2. Тип курса - coursetype
Нетиповой справочник
Потребление справочника
Данные справочника реплицируются
3. Вид страхования - instype
Простой справочник
Потребление справочника
Доступ к справочнику по API
4. Вид объекта страхования - insobjecttype Простой справочник
Потребление справочника
Доступ к справочнику по API
5. Вид страхового риска - insrisktype Простой справочник
Потребление справочника
Данные справочника реплицируются
6. Договор страхования - InsContract
Учётные данные
Ссылается на "Вид страхового риска"

Тестовые запросы

Микросервис "Валюта". Справочник "Тип валюты"

1. Получить по ИД

```
curl --location --request POST
'http://localhost:8081/currency/currencytype/getById' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":1
}'
```

2. Получить весь список

```
curl --location --request POST
'http://localhost:8081/currency/currencytype/listAll'
```

Микросервис "Валюта". Справочник "Валюта"

1. Получить по ИД

```
curl --location --request POST 'http://localhost:8081/currency/currency/getById' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":6
}'
```

2. Получить весь список

```
curl --location --request POST 'http://localhost:8081/currency/currency/listAll'
```

3. Добавить

```
curl --location --request POST 'http://localhost:8081/currency/currency/add' \
--header 'Content-Type: application/json' \
--data-raw '{
    "name":"Тугрик",
    "brief":"TUG",
    "code":"998",
    "prec":2,
    "currencyType": {
    "id": {
            "id": 2,
            "discriminator": "CURRENCYTYPE"
        },
        "name": "Дополнительная",
        "brief": "Дополнительная",
        "code": "Extended",
        "deleted": false
    }
}'
```

4. Изменить

```
curl --location --request POST 'http://localhost:8081/currency/currency/update' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":212106,
    "name":"TUG222"
}'
```

5. Удалить

```
curl --location --request POST 'http://localhost:8081/currency/currency/delete' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":212106
}'
```

Микросервис "Валюта". Справочник "Тип курса"

1. Получить по ИД

```
curl --location --request POST 'http://localhost:8081/currency/coursetype/getById'
\
--header 'Content-Type: application/json' \
--data-raw '{
    "id":8
}'
```

2. Получить весь список

```
curl --location --request POST 'http://localhost:8081/currency/coursetype/listAll'
```

3. Добавить

```
curl --location --request POST 'http://localhost:8081/currency/coursetype/add' \
--header 'Content-Type: application/json' \
--data-raw '{
    "name":"Курс ППП",
    "brief":"КурсППП"
}'
```

4. Изменить

```
curl --location --request POST 'http://localhost:8081/currency/coursetype/update' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": 212104,
    "name": "Курс PPP",
    "brief": "КурсPPP"
}'
```

5. Удалить

```
curl --location --request POST 'http://localhost:8081/currency/coursetype/delete' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":212104
}'
```

Микросервис "Валюта". Справочник "Курс"

1. Получить по ИД

```
curl --location --request POST 'http://localhost:8081/currency/course/getById' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":1
}'
```

2. Получить весь список

```
curl --location --request POST 'http://localhost:8081/currency/course/listAll'
```

3. Добавить

```
curl --location --request POST 'http://localhost:8081/currency/course/add' \
--header 'Content-Type: application/json' \
--data-raw '{
    "date": 1609621200000,
    "courseType": {
        "id": 8
    },
    "fromCurrency": {
        "id": 6
    },
    "toCurrency": {
        "id": 7
    },
    "rate": 77.0
}'
```

4. Изменить

```
curl --location --request POST 'http://localhost:8081/currency/course/update' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": 212105,
    "date": 1609621200000,
    "courseType": {
        "id": 8
    },
    "fromCurrency": {
        "id": 6
    },
    "toCurrency": {
        "id": 7
    },
    "rate": 77.5
}'
```

5. Удалить

```
curl --location --request POST 'http://localhost:8081/currency/course/delete' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":212105
}'
```

Микросервис "Страховые продукты". Справочник "Вид объекта страхования"

1. Получить по ИД

```
curl --location --request POST
'http://localhost:8082/insproduct/insobjecttype/getById' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":12
}'
```

2. Получить весь список

```
curl --location --request POST
'http://localhost:8082/insproduct/insrisktype/listAll'
```

3. Получить упорядоченный список

```
curl --location --request POST
'http://localhost:8082/insproduct/instype/orderedListAll' \
--header 'Content-Type: application/json' \
--data-raw '{
    "orderBy": "brief asc, name desc"
}'
```

4. Получить список по условиям

```
curl --location --request POST
'http://localhost:8082/insproduct/insrisktype/findAll' \
--header 'Content-Type: application/json' \
--data-raw '{
    "where": "brief like :brief",
    "parameters" :{
        "brief": "%2%"
    },
    "orderBy": "name desc"
}'
```

5. Добавить

```
curl --location --request POST 'http://localhost:8082/insproduct/insrisktype/add' \
--header 'Content-Type: application/json' \
--data-raw '{
    "name":"Ответственность 3",
    "brief":"Ответственность3"
}'
```

6. Изменить

```
curl --location --request POST
'http://localhost:8082/insproduct/insrisktype/update' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":47,
    "name":"Ответственность 444",
    "brief":"Ответственность444"
}'
```

7. Удалить

```
curl --location --request POST
'http://localhost:8082/insproduct/insrisktype/delete' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":44
}'
```

Микросервис "Страховые продукты". Данные нескольких справочников

1. Получить весь список

```
curl --location --request POST 'http://localhost:8082/insproduct/refbooks/listAll'
\
--header 'Content-Type: application/json' \
--data-raw '{
  "aliases": ["instype", "insobjecttype", "insrisktype"]
}'
```

2. Получить упорядоченный список

```
curl --location --request POST
'http://localhost:8082/insproduct/refbooks/orderedListAll' \
--header 'Content-Type: application/json' \
--data-raw '{
    "refbooks": [
        {
            "alias": "instype",
            "orderBy": "id desc"
        },
        {
            "alias": "insobjecttype",
            "orderBy": "id desc"
        },
        {
            "alias": "insrisktype",
            "orderBy": "id desc"
        }
    ]
}'
```

3. Получить список по условиям

```
curl --location --request POST 'http://localhost:8082/insproduct/refbooks/findAll'
\
--header 'Content-Type: application/json' \
--data-raw '{
    "refbooks": [
        {
            "alias": "instype",
            "where": "brief like :brief",
            "parameters" :{
                "brief": "%o%"
            },
            "orderBy": "name desc"
        },
        {
            "alias": "insobjecttype",
            "where": "brief like :brief",
            "parameters" :{
                "brief": "%C%"
            },
            "orderBy": "id desc"
        },
        {
            "alias": "insrisktype",
            "where": "brief like :brief",
            "parameters" :{
                "brief": "%2%"
            },
            "orderBy": "id desc"
        }
    ]
}'
```

Микросервис "Учёт договоров страхования". "Договоры страхования"

1. Получить по ИД

```
curl --location --request POST 'http://localhost:8083/inscontract/getById' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id":1
}'
```

2. Получить весь список

```
curl --location --request POST 'http://localhost:8083/inscontract/listAll'
```