Основной модуль JPrime

Содержит типовую логику, API и базовые реализации основных функций системы

Описание

Содержит логику работы с метой + основные интерфейсы

- Загружает метаописание из xml в момент старта
- Предоставляет базовое
- Загружает маппинг метаописания из xml в момент старта

Содержимое ресурсов

- jwt - для публичных ключей проверки JWT
- meta - для xml с метаописанием
- metamaps - для xml с маппингом метаописания

Системные типы

Код	Java тип	Описание
backReference	-	Обратная ссылка
biginteger	BigInteger	Длинное целочисленное
boolean	Boolean	Да/Нет
datFe	LocalDate	Дата
dateRange	JPDateRange	Диапазон дат
datetime	LocalDateTime	Полная дата (без учета часового пояса)
double	Double	Вещественное (64 бита)
file	String	Файл
float	Float	Вещественное (32 бита)
integer	Integer	Целочисленное (32 бита)
int4range	JPIntegerRange	Диапазон целочисленный (32 бита)
json	String	JSON
long	Long	Целочисленное (64 бита)
money	JPMoney	Денежное
simpleFraction	JPSimpleFraction	Простая дробь

Код	Java тип	Описание
string	String	Строка
timestamp	LocalDateTime	Полная дата (с учетом часового пояса)
time	LocalTime	Время
tsRange	JPDateTimeRange	Диапазон дат со временем
uuid	UUID	Глобальный идентификатор
xml	String	XML-элемент
virtualReference	-	Виртуальное значение

Виртуальное значение

Код	Java тип	Описание
virtualReference	-	Глобальный идентификатор

Виртуальное значение позволяет отобразить значение связанного объекта, как свойство текущего объекта Для построения виртуальной ссылки необходима прямая ссылка на класс, свойство которого хотим виртуализировать

Для полноценной работы виртуальной ссылки необходимо также указать `virtualReference` (путь виртуальной ссылки) + `virtualType` (тип итогового значения виртуальной ссылки)

Пример:

`sprWorker`

Код	Описание
ouid	Идентификатор
fio	Фамилия/Имя/Отчество

`sprOrg`

Код	Описание
ouid	Идентификатор
name	Название
director	Ссылка на sprWorker.ouid
directorFIO	Виртуальное значение на sprWorker.fio

, где

```
<jpAttr><code>directorFIO</code><type>virtualReference</type><virtualReference>director.fio</virtualR
```

Глобальный идентификатор

Код	Java тип	Описание
uuid	java.util.UUID	Глобальный идентификатор

Денежное

Физически в атрибуте типа `денежное` хранится размер, но для полноценной работы атрибута необходимо корректное заполнение `money` свойства ,где

Код	Описание	Обязательное
currencyCode	Код валюты	+

пример через xml

```
      <jpAttr>
        <guid>69269c0d-e8a6-49bc-aebe-f71e4242640c</guid>
        <code>moneyAttr</code>
        <type>money</type>
        <qName>jpClass.className1.PersonalCard.moneyAttr</qName>
        <description>Деньги</description>
        <money>
          <currencyCode>RUR</currencyCode>
        </money>
      </jpAttr>
```

Обратная ссылка

Код	Java тип	Описание
backReference	-	Глобальный идентификатор

Обратная ссылка не является полноценным свойством объекта: у него нет значения, хранящегося отдельно. Это, скорее визуальная настройка, позволяющая вывести все объекты другого класса, связанные с текущим объектом

Для полноценной работы обратной ссылки необходимо указать `refJpClass` и `refJpAttr` (кодовое имя класса и кодовое имя атрибута соответственно), где указанный `refJpAttr` - является прямой ссылкой на идентификатор текущего класса

Простая дробь

Физически в атрибуте типа `простая дробь` хранится числетель, а для полноценной работы атрибута необходимо корректное заполнение `simpleFraction` свойства ,где

Код	Описание	Обязательное
integerAttrCode	Кодовое имя атрибута для хранения целого значения	опционально

denominatorAttrCode	Кодовое имя атрибута для хранения знаменателя	опционально
Код	**Описание**	**Обязательное**

пример через xml

```
      <jpAttr>
        <guid>69269c0d-e8a6-49bc-aebe-f71e4142640c</guid>
        <code>simpleFract</code>
        <type>simpleFraction</type>
        <qName>jpClass.className1.PersonalCard.simpleFract</qName>
        <description>Доля</description>
        <simpleFraction>
          <integerAttrCode>integerAttr</integerAttrCode>
          <denominatorAttrCode>denominatorAttr</denominatorAttrCode>
        </simpleFraction>
      </jpAttr>
```

Прямая ссылка (Nx1)

Прямая ссылка не реализуется отдельным типом, а расширяет свойства атрибута полями `refJpClass` и `refJpAttr` Тип атрибута указывается один из простых (строка, целочисленное и т.д.), важно, что тип поля соответствовал типу атрибута класса, указанного в настройках `refJpClass` и `refJpAttr` (кодовое имя класса и кодовое имя атрибута соответственно)

Строка

Файл

Физически в атрибуте типа `файл` хранится имя файла в целевом хранилище, а для полноценной работы атрибута необходимо корректное заполнение `refJpFile` свойства ,где

Код	Описание	Обязательное
storageCode	Код ФС хранилища с файлом	+
storageFilePath	Путь в ФС хранилище с файлом	+
storageCodeAttrCode	Кодовое имя атрибута для хранения кода ФС хранилища с файлом	опционально
storageFilePathAttrCode	Кодовое имя атрибута для хранения пути в ФС хранилище с файлом	опционально
fileTitleAttrCode	Кодовое имя атрибута для хранения исходного имени файла	опционально
fileExtAttrCode	Кодовое имя атрибута для хранения расширения файла	опционально
fileSizeAttrCode	Кодовое имя атрибута для хранения размера файла (в байтах)	опционально
fileDateAttrCode	Кодовое имя атрибута для хранения даты формирования файла	опционально

Код	Описание	Обязательное
fileInfoAttrCode	Кодовое имя атрибута для хранения дополнительной информации о файле	опционально

пример через xml

```
      <jpAttr>
        <code>scanCopy</code>
        <type>file</type>
        <qName>jpClass.className1.PersonalCard.scanCopy</qName>
        <description>Скан копия гражданина</description>
        <refJpFile>
          <storageCode>fileStorage</storageCode>
          <storageFilePath>filePath</storageFilePath>
          <storageCodeAttrCode>attr1</storageCodeAttrCode>
          <storageFilePathAttrCode>attr2</storageFilePathAttrCode>
          <fileTitleAttrCode>attr4</fileTitleAttrCode>
          <fileExtAttrCode>attr5</fileExtAttrCode>
          <fileSizeAttrCode>attr6</fileSizeAttrCode>
          <fileDateAttrCode>attr7</fileDateAttrCode>
          <fileInfoAttrCode>attr8</fileInfoAttrCode>
        </refJpFile>
      </jpAttr>
```

Преобразование java типов

Для преобразовавания T1 в T2 разных классов рекомендуется использовать `mp.jprime.parsers.ParserService`

```
  private ParserService parserService;

 @Autowired
 private void setParserService(ParserService parserService) {
   this.parserService = parserService;
 }
 ...

 private LocalDate getLocalDate(Object value) {
   if (value == null) {
     return null;
   }
   return parserService.parseTo(LocalDate.class, value);
 }
```

Само преобразование описывается наследником от `mp.jprime.parsers.TypeParser`. Причем справочник может расширятся, как в коде JPrime, так и в прикладном коде

Преобразование значений по типу атрибута

Для преобразовавания к значению, соответствующему типу атрибута, рекомендуется использовать `mp.jprime.attrparsers.AttrParserService`

```
 private AttrParserService attrParserService;

@Autowired
private void setPAttrParserService(AttrParserService attrParserService) {
  this.attrParserService = attrParserService;
}
...

private LocalDate getDate(JPAttr attr, JPAttrData data) {
  return attrParserService.parseTo(attr, data);
}
```

Само преобразование описывается наследником от `mp.jprime.attrparsers.AttrTypeParser`. Причем справочник может расширятся, как в коде JPrime, так и в прикладном коде

Метаописание

Описание структуры данных и их взаимосвязей. Позволяет статически или динамически описывать классы, их свойства (атрибуты) и связи между ними.

Структура меты

Описание класса `JPClass`

Свойство	Описание
guid	Глобальный идентификатор
code	Кодовое имя класса
qName	Полный код класса
name	Название класса
shortName	Короткое название класса
description	Описание класса
jpPackage	Настройки доступа
attrs	Список атрибутов класса

Описание атрибута класса `JPAttr`

Свойство	Описание
guid	Глобальный идентификатор
code	Кодовое имя атрибута
qName	Полный код атрибута

Свойство	Описание
name	Название атрибута
shortName	Короткое название атрибута
description	Описание атрибута
jpPackage	Настройки доступа
identifier	Признак идентификатора
mandatory	Признак обязательности
type	Тип атрибута
refJpClass	Кодовое имя класса, на который ссылается
refJpAttr	Кодовое имя атрибута ссылочного класса
refJpFile	Описание файлового атрибута
simpleFraction	Описание простой дроби
money	Описание денежного типа
virtualReference	Путь виртуальной ссылки
virtualType	Тип виртуальной ссылки
length	Длина (для строковых полей, в том числе для строковых виртуальных)
jpProps	Свойства псевдо-меты
schemaProps	Схемы вложенных свойств псевдо-меты

Описание свойства псевдо-меты `JPProperty`

Свойство	Описание
code	Кодовое имя свойства
qName	Полный код свойства
name	Название свойства
shortName	Короткое название свойства
description	Описание свойства
mandatory	Признак обязательности
multiple	Признак множественности

type	Тип свойства
Свойство	**Описание**
length	Длина (для строковых полей, в том числе для строковых виртуальных)
refJpClass	Кодовое имя класса, на который ссылается
refJpAttr	Кодовое имя атрибута ссылочного класса
schemaCode	Код схемы, описывающей структуру свойства

Описание схемы свойств псевдо-меты `JPPropertySchema`

Свойство	Описание
code	Кодовое имя свойства
jpProps	Свойства псевдо-меты

- При описании свойства `JPProperty` с типом `ELEMENT` необходимо так же описать его схему в свойстве `schemaProps` у `@JPAttr` с помощью `@JPPropertySchema` и указать код этой схемы в поле `schemaCode` аннотации `JPProperty`
- Схемы свойств можно переиспользовать в пределах одного атрибута.

Способы описания меты

Аннотации

- Аннотация детализирует наследника от `mp.jprime.meta.JPMeta`

  ``` @JPClass( guid = "cc53f898-ceec-49b3-81a3-43eb3fdc43f0", code = "userEvent", qName = "common.userEvent", name = "Таблица с информацией о пользовательских событиях", shortName = null, description = null, jpPackage = "adminAccess", attrs = { @JPAttr( guid = "a6f40c06-e41f-4003-a4df-a72d304adc7e", code = "eventGuid", type = JPType.UUID, identifier = true, qName = "common.userevent.eventGuid", name = "Гуид события", shortName = null, description = null
  ), @JPAttr( guid = "a6f40c06-e41f-4003-a4df-a72d304adc7d", code = "event", type = JPType.JSON, qName = "common.userevent.event", name = "JSON-описание события", shortName = null, description = null, jpProps = { @JPProperty( code = "event_desc", type = PropertyType.ELEMENT, qName = "common.userevent.prop", name = "JSON-описание события", shortName = null, description = null, schemaCode = "test_schema" ) }, schemaProps = { @JPPropertySchema( code = "test_schema", jpProps = { @JPProperty( code = "event_string", type = PropertyType.STRING, qName = "common.userevent.string", name = "поле-описание события", shortName = null, description = null ) } ) }
  ) } )

```
#### XML

* xml файл, указанной структуры, должен быть размещен в src/main/resources/meta
```

cc53f898-ceec-49b3-81a3-43eb3fdc43f0 `userEvent` common.userEvent Таблица с информацией о пользовательских событиях a6f40c06-e41f-4003-a4df-a72d304adc7e `eventGuid` uuid 0 true common.userevent.eventGuid Гуид события `testprop` Test Prop egu.eguApplication.jsonattr.jsonprop element `innertestprop` Inner Test Prop egu.eguApplication.jsonattr.injsonprop string
```

Типовые атрибуты

В JPrime существует набор системных зарезервированных кодовых имен атрибутов

Кодовое имя атрибута	Описание	Примечание
userOwnerId	Пользователь, создавший объект	Заполняется userId при создании
creationDate	Дата создания	Заполняется датой создания
userEditorId	Пользователь, изменивший объект	Заполняется userId при создании и изменении
changeDate	Дата редактирования	Заполняется датой создания и изменения
jpPackage	Настройка доступа к объекту	Содержит код настройки доступа к объекту

Описание маппинга

Настройка хранения данных в определенном хранилище

Структура маппинга

Описание класса ``JPClassMap``

Свойство	Описание
code	Кодовое имя класса
storage	Кодовое имя хранилища
map	Мап на хранилище
schema	Схема в хранилище
attrs	Список маппинга атрибутов класса

Описание атрибута класса ``JPAttrMap``

Свойство	Описание
code	Кодовое имя атрибута
map	Мап на хранилище
fuzzyMap	Мап на поле с индексами нечеткого поиска
cs	Регистр значений
readOnly	Запрет на изменение значений

Способы описания маппинга

Аннотации

* Аннотация детализирует наследника от ``mp.jprime.meta.JPMeta``

@JPClassMap(code = "userEvent", storage = "userevents", map = "userevent", attrs = { @JPAttrMap(code = "eventGuid", map = "eventguid") })

XML

* xml файл, указанной структуры, должен быть размещен в src/main/resources/metamaps

Настройки RBAC

Настройки доступа разрешительные и запретительные для определенных ролей

Структура настроек

Свойство	Описание
code	Уникальный код настроек доступа
access	Список настроек по ролям

Свойство	Описание
type	Тип настройки: разрешительный или на запрет
role	Кодовое имя роли
read	Доступ на чтение
create	Доступ на создание
update	Доступ на изменение
delete	Доступ на удаление

Способы описания настроек

Аннотации

* Аннотация детализирует наследника от ``mp.jprime.security.JPSecuritySettings``

@JPPackages({ @JPPackage(code = "commonAccess", access = { @JPAccess(type = JPAccessType.PERMIT, role = "AUTH_ACCESS", read = true, create = true, update = true, delete = true) }), @JPPackage(code = "commonDenied", access = { @JPAccess(type = JPAccessType.PROHIBITION, role = "AUTH_ACCESS", read = true, create = true, update = true, delete = true) }) })

XML

* xml файл, указанной структуры, должен быть размещен в src/main/resources/security

Настройки ABAC

Разграничение доступа доступа на основе атрибутов субъекта, окружения и ресурса

Структура настроек

Группа политик

Свойство	Описание
name	Название группы политики
qName	QName названия
jpClasses	Список классов, к которым применяется группа политик. Если список пуст, то ко всем данны
policies	Список политик

Политика

Свойство	Описание
name	Название политики
qName	QName названия
actions	Действия, к которым применяется политика (read, create, update, delete)
subjectRules	Список правил для пользователя
resourceRules	Список правил для объекта
environmentRules	Список правил для окружения

Правило для пользователя

Свойство	Описание
name	Название правила
qName	QName правила
effect	Возвращает тип доступа (разрешительный/запретительный) permit/prohibition
username	Условие на логин
role	Условие на роли
orgId	Условие на организацию
depId	Условие на подразделение

Правило для окружения

Свойство	Описание
name	Название правила
qName	QName правила
effect	Возвращает тип доступа (разрешительный/запретительный) permit/prohibition
ip	Условие на IP
time	Условие на время применения политики
time.daysOfWeek	Дни: пнд/вск
time.fromTime	Начало периода в формате HH:mm
time.toTime	Окончание периода в формате HH:mm
time.fromDateTime	Начало периода в формате yyyy-MM-dd'T'HH:mm:ss
time.toDateTime	Окончание периода в формате yyyy-MM-dd'T'HH:mm:ss

Правило для объекта

```
| Свойство | Описание                                                                    |
|----------|-----------------------------------------------------------------------------|
| name     | Название правила                                                            |
| qName    | QName правила                                                               |
| effect   | Возвращает тип доступа (разрешительный/запретительный) permit/prohibition |
| attr     | Кодовое имя атрибута                                                        |
| cond     | Условие на атрибут                                                          |

#### Условие

| Свойство | Описание        |
|----------|-----------------|
| in       | Список значений |
| notIn    | Список значений |

### Способы описания настроек

#### Аннотации

* Аннотация детализирует наследника от ``mp.jprime.security.JPSecuritySettings``
```

@JPPolicySets(value = { @JPPolicySet(name = "Группа политик annotation-тест", qName = "jpPolicySet.annotationTest", jpClasses = {"main"}, policies = { @JPPolicy(name = "Политика Создание", qName = "jpPolicySet.annotationTest.create", actions = {JPAction.CREATE}, subjectRules = { @JPSubjectRule(name = "Доступно только роли TEST_ROLE", qName = "jpPolicySet.annotationTest.create.rule1", role = @JPCond(in = {"ADMIN_ROLE", "TEST1_ROLE"}), effect = JPAccessType.PERMIT) }, environmentRules = { @JPEnvironmentRule(name = "Доступно только в течении для 2010-01-10", qName = "jpPolicySet.annotationTest.create.envRule1", time = @JPTime(daysOfWeek = {DayOfWeek.MONDAY, DayOfWeek.THURSDAY}, fromTime = "10:00", toTime = "17:59", fromDateTime = "2010-01-10T00:00:00", toDateTime = "2020-01-10T23:59:59")) }), @JPPolicy(name = "Политика Обновление", qName = "jpPolicySet.annotationTest.update1", actions = {JPAction.READ, JPAction.UPDATE}, subjectRules = { @JPSubjectRule(name = "Доступно только роли ADMIN_ROLE", qName = "jpPolicySet.annotationTest.update1.rule1", role = @JPCond(in = {"ADMIN_ROLE"}), effect = JPAccessType.PERMIT) }), @JPPolicy(name = "Политика Обновление", qName = "jpPolicySet.annotationTest.update2", actions = {JPAction.READ, JPAction.UPDATE}, subjectRules = { @JPSubjectRule(name = "Доступно только роли TEST1_ROLE", qName = "jpPolicySet.annotationTest.update2.rule1", role = @JPCond(in = {"TEST1_ROLE"}), effect = JPAccessType.PERMIT) }, resourceRules = { @JPResourceRule(name = "Доступны test1 и test2", qName = "jpPolicySet.annotationTest.update2.rule1", attr = "name", cond = @JPCond(in = {"test1", "test2"}), effect = JPAccessType.PERMIT), @JPResourceRule(name = "Доступны все, кроме test2", qName = "jpPolicySet.annotationTest.update2.rule2", attr = "name", cond = @JPCond(notIn = {"test2"}), effect = JPAccessType.PERMIT), @JPResourceRule(name = "Не доступен test3", qName = "jpPolicySet.annotationTest.update3.rule3", attr = "name", cond = @JPCond(in = {"test3"}), effect = JPAccessType.PROHIBITION) }),}) })

```
#### XML

* xml файл, указанной структуры, должен быть размещен в src/main/resources/abac
```

Группа политик xml-тест jpPolicySet.xmlTest test1 test2 Политика Создание jpPolicySet.xmlTest.create create Доступно только роли TEST_ROLE jpPolicySet.xmlTest.create.rule1 TEST_ROLE TEST_ROLE permit Доступно в определенные часы jpPolicySet.xmlTest.create.environmentRule1 permit 6,7 10:00 17:59 2010-01-10T00:00:00 2020-01-10T23:59:59 Политика Обновление jpPolicySet.xmlTest.update read update Доступно только автору jpPolicySet.xmlTest.update.rule1 userOwnerId {AUTH_USERID} permit

```
## JPrime API

Взаимодействие с объектами происходит через метамодель и кодовые имена классов/атрибутов

### Чтение данных

Пример:
```

@Autowired private JPObjectRepository repo;

JPSelect jpSelect = JPSelect .from(jpClass.getCode()) .auth(authInfo) .where(Filter.attr(attrCode1).eq(value1)) .limit(limit) .offset(offset) .orderBy(attrCode2, OrderDirection.DESC) .build()

return repo.getAsyncList(jpSelect)

```
### Создание данных

Пример:
```

@Autowired private JPObjectRepository repo;

JPCreate jpCreate = JPCreate .create(jpClass.getCode()) .auth(authInfo) .set(attrCode1, value1) .set(attrCode2, value2) .build()

return repo.asyncCreate(jpCreate)

```
### Обновление данных

Пример:
```

@Autowired private JPObjectRepository repo;

JPUpdate jpUpdate = JPUpdate .update(JPId.get(jpClass.getCode(), objectId)) .auth(authInfo) .set(attrCode1, value1) .set(attrCode2, value2) .build()

return repo.asyncUpdate(jpUpdate)

```
### Удаление данных

Пример:
```

@Autowired private JPObjectRepository repo;

JPDelete jpDelete = JPDelete .delete(JPId.get(jpClass.getCode(), objectId)) .auth(authInfo) .build();

return repo.asyncDelete(jpDelete)

```
### Поиск данных

#### Критерии поиска
```

Свойство	Описание	backReference	bigint
isNull	Значение не указано	-	+
isNotNull	Значение указано	-	+
eq	Значение равно указанному	-	+
neq	Значение не равно указанному	-	+
gt	Значение больше указанного	-	+
gte	Значение больше или равно указанному	-	+
lt	Значение меньше указанному	-	+
lte	Значение меньше или равно указанному	-	+
eqYear	Значение равно указанному году	-	-
neqYear	Значение не равно указанному году	-	-
gtYear	Значение больше указанного года	-	-
gteYear	Значение больше или равно указанному года	-	-
ltYear	Значение меньше указанного года	-	-
lteYear	Значение меньше или равно указанного года	-	-
eqMonth	Значение равно указанному месяцу	-	-
neqMonth	Значение не равно указанному месяцу	-	-
gtMonth	Значение больше указанного месяца	-	-
gteMonth	Значение больше или равно указанному месяца	-	-
ltMonth	Значение меньше указанного месяца	-	-
lteMonth	Значение меньше или равно указанного месяца	-	-
eqDay	Значение равно указанному дню	-	-
neqDay	Значение не равно указанному дню	-	-
gtDay	Значение больше указанного дня	-	-
gteDay	Значение больше или равно указанному дню	-	-
ltDay	Значение меньше указанного дня	-	-
lteDay	Значение меньше или равно указанного дню	-	-
between	Значение находится в указанном диапазоне	-	+
exists	Ссылочный атрибут содержит объекты по указанному условию	+	-
notExists	Ссылочный атрибут не содержит объекты по указанному условию	+	-
like	Значение содержит указанное	-	-
fuzzyLike	Нечеткий поиск	-	-
fuzzyOrderLike	Нечеткий поиск с учетом порядка слов	-	-
startsWith	Начинается с указанного	-	-
notStartsWith	Не начинается с указанного	-	-
in	Значение находится в указанном списке	-	+
containsRange	Значение(диапазон) содержит указанный диапазон	-	-
containsEl	Значение(диапазон) содержит указанный элемент	-	-
overlapsRange	Значение(диапазон) содержится в указанном диапазоне	-	-
eqRange	Значение равно указаному диапазону	-	-
neqRange	Значение не равно указаному диапазону	-	-
gtRange	Значение больше указанного диапазона	-	-
gteRange	Значение больше или равно указанному диапазону	-	-
ltRange	Значение меньше указанного диапазона	-	-
lteRange	Значение больше или равно указанному диапазону	-	-

Дополнительно, для ссылочных атрибутов со свойствами `refJPClass`+`refJPAttr` так же можно использовать
для поиска по ссылочным объектам: `exists` и `notExists`

```
Значения поиска для `virtualReference` соответствуют конечному типу данных, указанному в 'virtualType'

## Настройки внешней публикации меты

`jprime.meta.api.filter.enabled` - признак публикации указанных данных
`jprime.meta.api.filter.jpClassCodes` - список кодовых имен классов через [,] для публикации
`jprime.meta.api.filter.jpStorageCodes` - список кодовых имен хранилищ через [,], классы которых публи

## Настройки доступа к мете анонимуса

`jprime.meta.api.filter.anonymous.enabled` - признак доступа анонимуса
`jprime.meta.api.filter.anonymous.jpClassCodes` - список кодовых имен классов через [,] для публикации

## Бины

Любые данные после получения из хранилища сериализуются в `mp.jprime.dataaccess.beans.JPObject`
При необходимости создать кастомный бин достаточно наследовать его от `mp.jprime.dataaccess.beans.JPOb
и указать аннотацией для каких классов он будет использоваться
```

@JPClassesLink(jpClasses = {Address.CLASS_CODE}) public class AddressBean extends JPObject {

}

```
Если на метаописание настроены два разных бина, один из которых является наследником другого, то для о

### Аннотация `mp.jprime.annotations.JPBeanInfo`

С помощью аннотации `mp.jprime.annotations.JPBeanInfo` можно:

#### Свойство `defaultJpAttrCollection`

Указывает набор атрибутов, которые будут всегда получаться из хранилища в случае ссылки на текущий кла

Пример:
```

@JPClassesLink(jpClasses = {Users.CLASS_CODE}) @JPBeanInfo(defaultJpAttrCollection = {Users.Attr.FULLNAME}) public class UsersBean extends JPObject {

}

````
Пояснение:

При наличии в любом классе атрибута, ссылающегося на users, при получении объекта этого класса, всегда

```json
{
    "id": 9970000303,
    "classCode": "companyJobFair",
    "data": {
        "..."
    },
    "linkedData": {
        "userOwnerId": {
            "id": 9999999912,
            "classCode": "users",
            "data": {
                "fullName": "Роман",
                "userId": 11111
            }
        }
    }
}
````

Валидаторы

Перед действиями над объектами есть возможность произвести проверки с помощью валидаторов:

- `mp.jprime.dataaccess.validators.JPClassValidator` для обычного использования
- `mp.jprime.dataaccess.validators.JPReactiveClassValidator` для реактивного кода

Вызов валидатора происходит перед всеми остальными действиями и по умолчанию производится перед открытием транзакции (если она не была принудительно открыта ранее)

Реализация валидатора на примере jprime.dataaccess.validators.JPClassValidator

Валидатор реализует указанный выше интерфейс, обычно является наследником класса `mp.jprime.dataaccess.validators.JPClassValidatorBase` и применяется к объектам класса, указанных в аннотации `JPClassesLink`

На один класс может быть настроено множество валидаторов, а один валидатор может применятся ко многим классам
````

```
/**
 * Проверка СНИЛС
 */
@JPClassesLink(
    jpClasses = {Contact.CLASS_CODE}
)
public class ContactValidator extends JPClassValidatorBase {
  @Override
  public void beforeCreate(JPCreate query) {
    ...
  }
}
```

В случае некорректных ситуаций реализация обычно выбрасывает JPRuntimeException

Если на метаописание настроены два разных валидатора, один из которых является наследником другого, то для обработки будет использоваться наследник

Вызов валидации в прикладном коде

Управлением всеми валидаторами осуществляется через `mp.jprime.dataaccess.validators.JPClassValidatorService`

```
   private JPClassValidatorService jpClassValidatorService;

  @Autowired
  private void setJPClassValidatorService(JPClassValidatorService jpClassValidatorService) {
    this.jpClassValidatorService = jpClassValidatorService;
  }

  ...

  private void validate(JPCreate query) {
    if (query == null) {
      return;
    }
    jpClassValidatorService.beforeCreate(query);
  }
```

Расчет значений по умолчанию

В JPrime есть возможность рассчитать значения по умолчанию с помощью `mp.jprime.dataaccess.defvalues.JPObjectDefValue`

Реализация расчета

Расчетный класс реализует указанный выше интерфейс и применяется к объектам класса, указанных в аннотации `JPClassesLink`

На один класс может быть настроено множество расчетных блоков, а один расчет может применятся ко многим классам

```
@JPClassesLink(
    jpClasses = {TestClass.CLASS_CODE}
)
public class TestClassDefValue implements JPObjectDefValue {
  @Override
  public void appendValues(JPMutableData jpData, JPObjectDefValueParams params) {
    jpData.putIfAbsent(TestClass.Attr.STATUS, TestClass.Status.STATUS_3);
  }
}
```

В случае некорректных ситуаций реализация обычно выбрасывает JPRuntimeException

Если на метаописание настроены два разных расчетных блока, один из которых является наследником другого, то для обработки будет использоваться наследник

Вызов расчета в прикладном коде

Управлением всеми расчетными блоками осуществляется через `mp.jprime.dataaccess.defvalues.JPObjectDefValueService`

```
   private JPObjectDefValueService jpObjectDefValueService;

  @Autowired
  private void setJPObjectDefValueService(JPObjectDefValueService jpObjectDefValueService) {
    this.jpObjectDefValueService = jpObjectDefValueService;
  }

  ...

  @Override
  public void beforeCreate(JPCreate query) {
    super.beforeCreate(query);
    query.getData().putIfAbsent(
        jpObjectDefValueService.getDefValues(query.getJpClass(), query.getAuth())
    );
  }
```

Дополнение значений

В JPrime есть возможность рассчитать значения атрибутов на основании имеющихся данных с помощью `mp.jprime.dataaccess.applyvalues.JPObjectApplyValue`

Реализация расчета

Расчетный класс реализует указанный выше интерфейс и применяется к объектам класса, указанных в аннотации `JPClassesLink`

На один класс может быть настроено множество расчетных блоков, а один расчет может применятся ко многим классам

```
@JPClassesLink(
    jpClasses = {TestClass.CLASS_CODE}
)
public class TestClassDefValue implements JPObjectApplyValue {
  @Override
  public JPMutableData getAppendValues(JPObjectApplyValueParams params) {
    return JPMutableData.of("attr1", LocalDate.now());
  }
}
```

В случае некорректных ситуаций реализация обычно выбрасывает JPRuntimeException

Если на метаописание настроены два разных расчетных блока, один из которых является наследником другого, то для обработки будет использоваться наследник

Вызов расчета в прикладном коде

Управлением всеми расчетными блоками осуществляется через `mp.jprime.dataaccess.applyvalues.JPObjectApplyValueService`

```
   private JPObjectApplyValueService jpObjectApplyValueService;

  @Autowired
  private void setJPObjectApplyValueService(JPObjectApplyValueService jpObjectApplyValueService) {
    this.jpObjectApplyValueService = jpObjectApplyValueService;
  }

  ...
    query.getData().putIfAbsent(
        jpObjectApplyValueService.getAppendValues(
          JPObjectApplyValueParamsBean.newBuilder(id, classCode, JPData.of(...))
                  .attrs(...)
                  .build()
        );
    );
```

Определение дополнительных сведений по объекту

В JPrime есть возможность возвратить набор информационных подсказок с помощью `mp.jprime.dataaccess.addinfos.JPObjectAddInfoService`

Поддержка разных форматов json для хранения и отображения

Реализация интерфейса `mp.jprime.attrparsers.jpjsonnode.JPJsonAttrValueConverter` позволяет использовать разные форматы для хранения и отображения (при конвертации в JsonJPObject)

Для этого необходимо реализовать класс, указав с помощью аннотации `mp.jprime.annotations.JPClassAttrsLink` атрибуты, для которых поддерживается конвертация

```
@JPClassAttrsLink(
    jpAttrs = {
        @JPClassAttr(
            jpClass = "jpClass1",
            jpAttr = "jsonAttr1"
        ),
        @JPClassAttr(
            jpClass = "jpClass2",
            jpAttr = "jsonAttr2"
        )
    }
)
public class JsonSignsConverter implements JPJsonAttrValueConverter {
  /**
   * Конвертирует данные из формата хранения в формат представления
   *
   * @param value Данные в формате хранения
   * @return Данные в формате представления
   */
  @Override
  public JPJsonNode toJsonView(JPJsonNode value) {
    // value = ... логика преобразования
    return value;
  }
}
```

Реализация источника данных

Источником данных являются реализации `mp.jprime.dataaccess.addinfos.JPObjectAddInfoProvider`, данных которых складываются

Источник данных - java код

Источник данных через java реализует интерфейс `mp.jprime.dataaccess.addinfos.JPObjectAddInfo` и применяется к объектам класса, указанных в аннотации `JPClassesLink`

На один класс может быть настроено множество реализаций, , данных которых в итоге складываются

```
@JPClassesLink(
    jpClasses = {TestClass.CLASS_CODE}
)
public class TestObjectAddInfo implements JPObjectAddInfo {
  @Override
  public Collection<AddInfo> getAddInfo(JPObjectAddInfoParams params) {
    Collection<AddInfo> result = new ArrayList<>();
    result.add(
        AddInfoBean.newBuilder().code("test1").info("Данные корректны").build()
    );
    return result;
  }
}
```

Источник данных - SQL

Одной из реализаций провайдера является определение дополнительных свойств через SQL запрос. Подробнее см. модуль `jprime-dataaccess-jdbc-addinfos`

Хендлера операций над объектами

Хендлер над операциями CRUD предоставляет собой реализацию интерфейса:

- `mp.jprime.dataaccess.handlers.JPClassHandler` для обычного использования
- `mp.jprime.dataaccess.handlers.JPReactiveClassHandler` для реактивного кода

  и позволяет обработать следущие события

```
  beforeCreate(JPCreate query);

  beforeUpdate(JPUpdate query);

  afterCreate(Comparable newObjectId, JPCreate query);

  afterUpdate(JPUpdate query);

  beforeDelete(JPDelete query);

  afterDelete(JPDelete query);
```

Связь java-хендлера и метаописания класса

Хендлер используется для обработки событий над объектами класса, указанных в аннотации `JPClassesLink`

```
@JPClassesLink(
    jpClasses = {PersBook.CLASS_CODE}
)
public class PersBookHandler extends JPClassHandlerBase {
  ...
}
```

При необходимости вызвать хендлер для всех объектов системы следует использовать следующий синтаксис

```
@JPClassesLink(
   uni = true
)
public class CommonHandler extends JPClassHandlerBase {
  ...
}
```

Если на метаописание настроены два разных хендлера, один из которых является наследником другого, то для обработки будет использоваться наследник

События

Все события в системе являются ассинхронными.

Системные события

События инициируются и обрабатываются программным кодом и используются для передачи информации между сервисами

Описание в модуле `jprime-common-core`

Пользовательские события

Событие является наследником класса `mp.jprime.events.userevents.JPUserEvent` и может быть инициировано в любом сервисе системы

События инициируются программным кодом и используются для отображении информации конечному пользователю

Инициализация события

```
 private UserEventPublisher eventPublisher;

@Autowired(required = false)
private void setUserEventPublisher(UserEventPublisher eventPublisher) {
  this.eventPublisher = eventPublisher;
}

...

private void publishEvent(Report r, AuthInfo authInfo) {
  if (eventPublisher == null) {
    return;
  }
  eventPublisher.publishEvent(JPUserEventData.newBuilder()
      .typeCode("report.created")
      .typeTitle("Отчет создан")
      .description("Отчет " + r.getFileTitle() + " сформирован")
      .objectClassCode("file")
      .objectId(r.getGuid())
      .userId(authInfo != null ? authInfo.getUserId() : null)
      .userDescription(authInfo != null ? authInfo.getUsername() : null)
      .build()
  );
}
```

Реализация UserEventPublisher

Существует две реализации передачи системных событий

- `KafkaUserEventPublisher`. Поддержка обмена событиями через Kafka.

Включается опцией `jprime.events.userevents.kafka.enabled=true` и рекомендуется для микросервисных сборок

- `ApplicationUserEventPublisher`. Поддержка обмена событиями внутри приложения

Включается опцией `jprime.events.userevents.app.enabled=true` и рекомендуется для монолитных сборок

Авторизация

Данные о пользователе предоставлены реализацией `mp.jprime.security.AuthInfo`, содержащей информацию о идентикаторе пользователя, его имени, организации и подразделении

Использование параметров авторизации в системе

В различных частях системы, например, блоке filter в search-запросах, возможно использование данных текущей авторизации

Свойство	Описание
AUTH_USERID	Идентификатор пользователя
AUTH_ORGID	Организация пользователя

Свойство	Описание
AUTH_DEPID	Подразделение пользователя
CUR_DATE	Текущий день
CUR_DATETIME	Текущее дата+время

Пример:

```
{
    "totalCount": true,
    "offset": 0,
    "limit": 50,
    "filter": {
        "and": [
            {
                "cond": {
                    "attr": "org",
                    "eq": "{AUTH_ORGID}"
                }
            }
        ]
    }
}
```