Руководство по использованию

Создание class-based аннотаций

Подготовка

Для создания аннотации, нужно:

- Получить билдер из класса ClassGenerator и передать в него имя класса, который мы хотим получить:

```
ClassGenerator.newBuilder("uiaChatMessageMeta")
```

- Добавить JavaDoc для класса:

```
.javaDoc("Внутренняя коммуникация. Сообщение чата пользователей")
```

- (Опционально) Добавить родителя (но не более одного, остальные будут проигнорированы):

```
.parent(JPMeta.class)
```

- (Опционально) Добавить вложенный интерфейс, чтобы была возможность добавлять константы для вложенных аннотаций:

```
.setInterface(innerInterfaceSetting())
```

- Добавить аннотацию (можно несколько, но желательно не добавлять несколько одинаковых аннотаций:

```
.annotation(mainAnnotation(), jpClass, innerAnnotation())
```

- Модификации (можно вызвать и оставить пустым):

```
.modifier()
```

- Собрать класс:

```
.build()
```

- Получить инпутстрим:

```
.toStream()
```

Обязательные настройки

- Для создания интерфейса обязательно создать настройки. Для этого нужно собрать InnerInterfaceSetting. Пример использования:

  ``` InnerInterfaceSetting.newBuilder() .innerInterfaceName("Attr") .innerInterfaceParent(JPMeta.Attr.class) .innerInterfaceJavaDoc("Атрибутивный состав") .build();
  ```

```
* Для создания аннотации обязательно создать настройки. Для этого нужно собрать MainAnnotationSetting
```

```
### MainAnnotationSetting
```

```
(Опционально) Настройки константы
```

ConstantSetting.newBuilder() .constantField("code") .constantName("CLASS_CODE") .javaDoc("Кодовое имя класса") .build();

```
(Опционально) Настройки дупликатов
```

DuplicatedFieldSetting.newBuilder() .mainField("name") .duplicated("shortName", "desctiption") .build();

```
Итоговая сборка главной аннотации
```

MainAnnotationSetting.newBuilder() .annotation(mp.jprime.meta.annotations.JPClass.class) .constants(constantSetting) .duplicatedFields(duplicatedFieldSetting) .build();

```
### InnerAnnotationSetting
```

```
(Опционально) Настройка констант
```

InnerConstantSetting.newBuilder() .constantField("code").javaDocField("name") .build();

```
(Опционально) Добавление вложенной аннотации во вложенную аннотацию
```

InnerAnnotationSetting.newBuilder() .annotation(mp.jprime.meta.annotations.JPFile.class) .nameField("refJpFile") .build();

```
Итоговая сборка вложенной аннотации
```

InnerAnnotationSetting.newBuilder() .annotation(mp.jprime.meta.annotations.JPAttr.class) .constants(innerConstantSetting) .nameField("attrs") //поле, в котором должна лежать аннотация .innerAnnotation(innerInnerAnnotationSetting) .build();

```
### Полный пример использования
```

ClassGenerator.newBuilder("uiaChatMessageMeta") .javaDoc("Внутренняя коммуникация. Сообщение чата пользователей") .parent(JPMeta.class) .setInterface(innerInterfaceSetting()) .annotation(mainAnnotation(), buildTestJPClass(), innerAnnotation()) .annotation(mainAnnotationMap(), buildJPClassMap(), innerAnnotationMap()) .modifier() .build() .toStream();

```
### Возможный результат
```

import java.lang.String; import mp.jprime.meta.JPMeta; import mp.jprime.meta.annotations.JPAttr; import mp.jprime.meta.annotations.JPClass; import mp.jprime.meta.annotations.JPFile; import mp.jprime.meta.beans.JPType; import mp.jprime.metamaps.annotations.JPAttrMap; import mp.jprime.metamaps.annotations.JPClassMap;

/** * Внутренняя коммуникация. Сообщение чата пользователей */ @JPClass(code = UiaChatMessageMeta.CLASS_CODE, name = "Внутренняя коммуникация. Сообщение чата пользователей", qName = "common.uia.uiaChatMessageMeta", jpPackage = "commonDenied", inner = true, actionLog = false, attrs = { @JPAttr(code = UiaChatMessageMeta.Attr.DATETIME, type = JPType.TIMESTAMP, name = "Дата сообщения", qName = "common.uia.uiaChatMessageMeta.dateTime"), @JPAttr(code = UiaChatMessageMeta.Attr.OUID, type = JPType.LONG, identifier = true, mandatory = true, name = "Идентификатор", qName = "common.uia.uiaChatMessageMeta.ouid", refJpFile = @JPFile(storageCode = "bvc", storageFilePath = "asdas", fileExtAttrCode = "dsfsd", fileDateAttrCode = "dfds", fileInfoAttrCode = "dfdsss")), @JPAttr(code = UiaChatMessageMeta.Attr.CHAT, type = JPType.STRING, name = "Идентификатор чата", qName = "common.uia.uiaChatMessageMeta.chat"), @JPAttr(code = UiaChatMessageMeta.Attr.AUTHOR, type = JPType.STRING, name = "Автор сообщения", qName = "common.uia.uiaChatMessageMeta.author"), @JPAttr(code = UiaChatMessageMeta.Attr.MESSAGE, type = JPType.STRING, name = "Текст сообщения", qName = "common.uia.uiaChatMessageMeta.message") }) @JPClassMap(code = UiaChatMessageMeta.CLASS_CODE, storage = "userinteraction", map = "uia_chat_message", attrs = { @JPAttrMap(code = UiaChatMessageMeta.Attr.DATETIME, map = "datetime"), @JPAttrMap(code = UiaChatMessageMeta.Attr.OUID, map = "ouid"), @JPAttrMap(code = UiaChatMessageMeta.Attr.CHAT, map = "chat"), @JPAttrMap(code = UiaChatMessageMeta.Attr.AUTHOR, map = "author"), @JPAttrMap(code = UiaChatMessageMeta.Attr.MESSAGE, map = "message") }) public final class UiaChatMessageMeta extends JPMeta { /** * Кодовое имя класса */ public static final String CLASS_CODE = "uiaChatMessageMeta";

/** * Атрибутивный состав */ public interface Attr extends JPMeta.Attr { /** * Дата сообщения */ String DATETIME = "dateTime";

```
 /**
  * Идентификатор
  */
 String OUID = "ouid";

 /**
  * Идентификатор чата
  */
 String CHAT = "chat";

 /**
  * Автор сообщения
  */
 String AUTHOR = "author";

 /**
  * Текст сообщения
  */
 String MESSAGE = "message";
```

}}

```
## Создание xml-based аннотаций

### Подготовка

Для создания аннотации, нужно:

* Получить билдер из класса XmlGenerator:
```

XmlGenerator.newBuilder()

```
* Добавить вложенную ноду (можно несколько, но желательно не добавлять несколько одинаковых нод:
```

.mainNode(mainNodeSetting(), buildTestJPClass(), innerNodeSetting())

```
* Собрать класс:
```

.build()

```
* Получить инпутстрим:
```

.toStream()

```
### Обязательные настройки

* Для создания аннотации (ноды) обязательно создать настройки. Для этого нужно собрать MainNodeSetting

### MainNodeSetting

(Опционально) Настройки дупликатов
```

DuplicatedFieldSetting.newBuilder() .mainField("name") .duplicated("shortName", "description") .build();

```
Итоговая сборка главной аннотации
```

MainNodeSetting.newBuilder() .annotation(mp.jprime.meta.annotations.JPClass.class)
.duplicatedFields(duplicatedFieldSetting) .name("JPClass") // Имя ноды .build();

```
### InnerNodeSetting

(Опционально) Добавление вложенной ноды во вложенную ноду
```

InnerNodeSetting.newBuilder() .annotation(mp.jprime.meta.annotations.JPFile.class) .nameField("refJpFile") .nameNode("JPFile") .build();

```
Итоговая сборка вложенной аннотации
```

InnerNodeSetting.newBuilder() .annotation(mp.jprime.meta.annotations.JPAttr.class) .nameField("attrs") // Поле, в котором должна лежать нода .nameNode("JPAttr") // Имя ноды .innerAnnotation(innerInnerAnnotationSetting) .build();

```
### Полный пример использования
```

Objects.requireNonNull(XmlGenerator.newBuilder()) .mainNode(mainNodeSetting(), buildTestJPClass(), innerNodeSetting()) .mainNode(mainNodeSettingMap(), buildJPClassMap(), innerNodeSettingMap()) .build() .toStream();

```
### Возможный результат
```

`uiaChatMessageMeta` Внутренняя коммуникация. Сообщение чата пользователей common.uia.uiaChatMessageMeta commonDenied true false `dateTime` timestamp Дата сообщения Дата сообщения Дата сообщения common.uia.uiaChatMessageMeta.dateTime `ouid` long true true Идентификатор Идентификатор Идентификатор common.uia.uiaChatMessageMeta.ouid bvc asdas dsfsd dfds dfdsss `chat` string Идентификатор чата Идентификатор чата Идентификатор чата common.uia.uiaChatMessageMeta.chat `author` string Автор сообщения Автор сообщения Автор сообщения common.uia.uiaChatMessageMeta.author `message` string Текст сообщения Текст сообщения Текст сообщения common.uia.uiaChatMessageMeta.message `uiaNotificationMeta` userinteraction uia_chat_message `dateTime` datetime `ouid` ouid `chat` chat `author` author `message` message ```