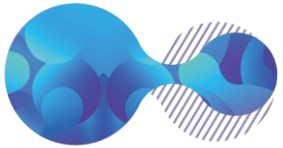
不同数据库类型JPA主键生成策略适配

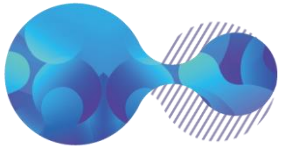
2020-01



1. 背景描述

@GeneratedValue的四种ID生成策略

策略	说明	特点
TABLE	使用关系型数据库中的一个表 (hibernate_sequence) 来生成主键	可移植性比较好，因为所有的关系型数据库都支持这种策略，但是性能差，不推荐使用
SEQUENCE	使用数据库序列生成主键	Oralce，DB2支持,MySQL不支持
IDENTITY	用一个 Identity 列来生成主键（主键自增）	MySQL支持，Oracle和DB2不支持
AUTO	将主键生成的策略交给持久化引擎 来决定，由它自己选择从 Table 策略，Sequence 策略和 Identity 策略三种策略中选择合适的主键生成策略	Oracle，DB2默认使用Sequence，MySQL默认使用Identity。 （Hibernate5.0之后MySQL下默认使用TABLE策略）

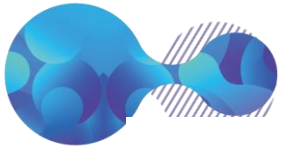


2. 问题描述

1.项目要求：同时适配MySQL，Oracle， DB2数据库

2.存在问题：使用GenerationType.AUTO策略在MySQL数据库下，使用hibernate_sequence表生成主键ID，所有表的ID都是基于hibernate_sequence递增，且存在性能问题。在Oracle，DB2数据库下，所有表的ID基于hibernate_sequence序列生成。

3.解决方案：MySQL数据库下每个表使用IDENTITY策略生成ID；Oracle，DB2等支持序列的数据库下使用SEQUENCE策略生成ID，并能够方便的切换到不同类型的数据库。



3. 技术原理

11.1 Annotations for Object/Relational Mapping

These annotations and types are in the package `javax.persistence`.

XML metadata may be used as an alternative to these annotations, or to override or augment annotations, as described in Chapter 12.

12.1 Use of the XML Descriptor

The XML schema for the object relational/mapping descriptor is contained in Section 12.3. The root element of this schema is the `entity-mappings` element. The absence or present of the `xml-mapping-metadata-complete` subelement contained in the `persistence-unit-defaults` subelement of the `entity-mappings` element controls whether the XML object/relational mapping descriptor is used to selectively override annotation values or whether it serves as a complete alternative to Java language metadata annotations.

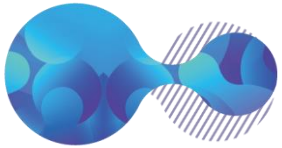
If the `xml-mapping-metadata-complete` subelement is specified, the complete set of mapping metadata for the persistence unit is contained in the XML mapping files for the persistence unit, and any persistence annotations on the classes are ignored.

4.1 解决方案（Oracle，DB2）

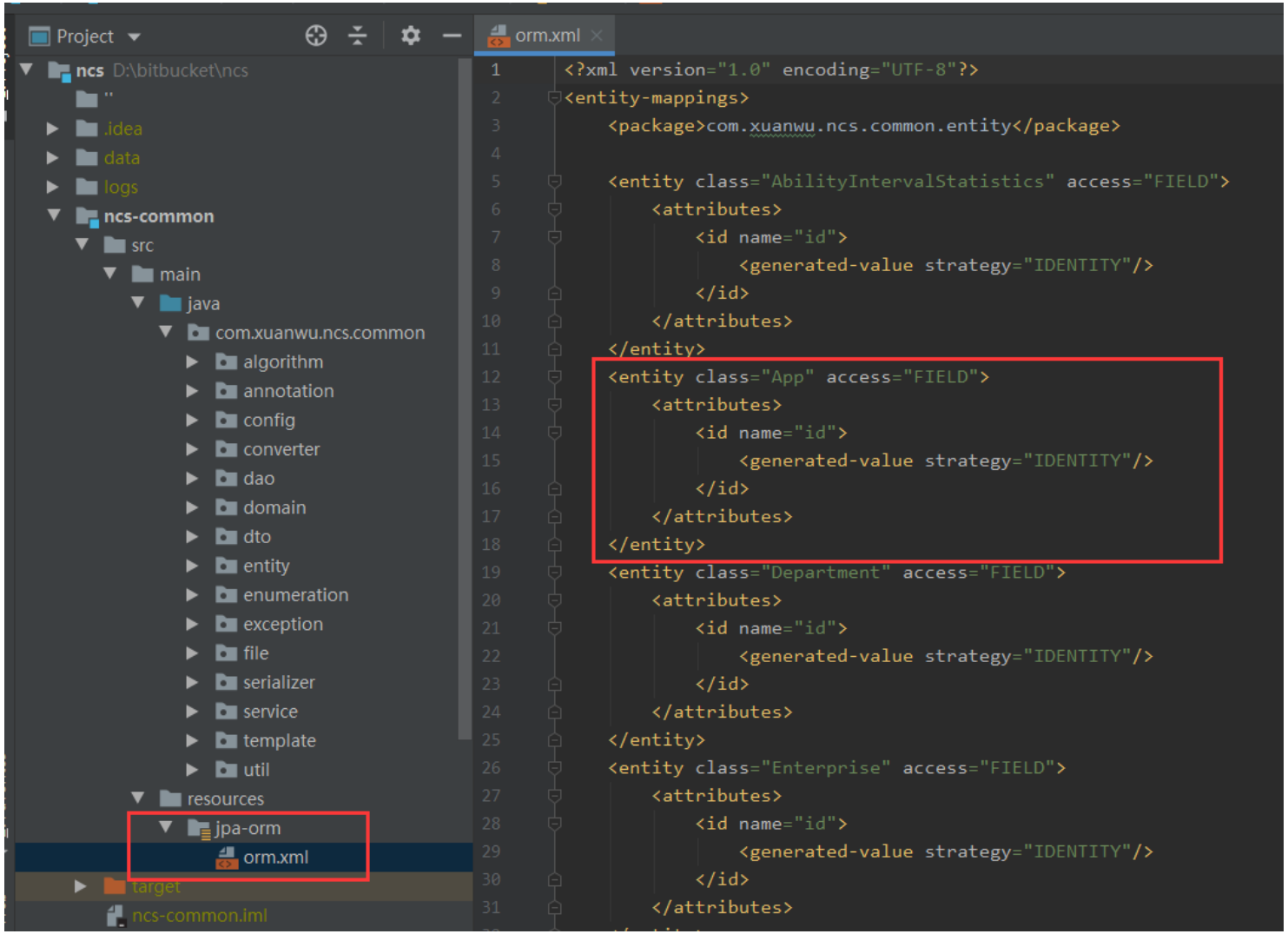
App.java

```java
package com.xuanwu.ncs.common.entity;

import ...

/**
 * 应用
 * @author ChenZhenQing
 * @date 2019/11/11 9:44
 */
@Data
@Entity
@Table(name = "ncs_app", uniqueConstraints = {
        @UniqueConstraint(name="INDEX_APP", columnNames = {"app_id", "app_key", "app_secret"})
})
public class App implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "ncs_app_seq")
    @SequenceGenerator(name = "ncs_app_seq", allocationSize = 1)
    private Integer id;

    /** 名称 */
    @Column(name = "name", length = 100)
    private String name;

    /** 包名 */
    @Column(name = "package_name", length = 100)
    private String packageName;

    /** 签名 */
    @Column(name = "signature", length = 32)
    private String signature;
```

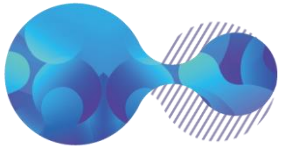
实体类中注解配置使用SEQUENCE序列生成ID



4.2 解决方案（MySQL）



配置文件中使用IDENTITY策略，覆盖注解配置



4.3 两侧策略切换方式

pom.xml (ncs-common)

```
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>${poi.version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>${poi.version}</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>easyexcel</artifactId>
            <version>${easyexcel.version}</version>
        </dependency>
    </dependencies>
    <build>
        <resources>
            <resource>
                <directory>src/main/resources/${orm}</directory>
                <targetPath>META-INF/</targetPath>
            </resource>
        </resources>
    </build>

</project>
```

pom.xml (ncs)

```
        </properties>
    </profile>
    <!--database-->
    <profile>
        <id>mysql</id>
        <activation>
            <activeByDefault>true</activeByDefault>
        </activation>
        <properties>
            <orm>jpa-orm</orm>
            <database>mysql</database>
        </properties>
    </profile>
    <profile>
        <id>oracle</id>
        <properties>
            <orm>none</orm>
            <database>oracle</database>
        </properties>
    </profile>
    <profile>
        <id>db2</id>
        <properties>
            <orm>none</orm>
            <database>db2</database>
        </properties>
    </profile>
</profiles>

<repositories>
    <repository>
```

project › profiles › profile

Maven

- Profiles
  - db2
  - dev
  - mysql
  - nexus
  - oracle
  - prod
  - sit
  - uat
- ncs (root)
- ncs-common
- ncs-init
- ncs-rest
- ncs-server
- ncs-statistics

4.4 项目打包

四、数据库ID生成策略适配

4.1支持数据库类型:

- MySQL数据库自增（默认）
- Oracle数据库序列
- DB2数据库序列

4.2技术原理：

hibernate支持数据库序列和数据库自增主键生成ID，可以在数据库表所对应的的实体类中通过@GeneratedValue设置ID生成策略， 但在不同的数据库类型下，ID生成策略配置不同，而注解只能设置固定类型的ID生成策略，为适配不同类型数据库，通过在META-INF 下添加orm.xml文件去覆盖实体类中的@GeneratedValue注解来设置其它类型的ID生成策略。

4.3项目配置

1.项目实体类中注解配置使用数据库序列生成ID，在ncs-common模块的src/main/resource/jpa-orm新增了orm.xml配置文件，该配 置文件使用数据库自增的方式覆盖了实体类中的数据库序列方式。

2.后续开发中新增数据库表需要在orm.xml添加新表的配置来做适配，否则在mysql下，新增的表将采用序列（MySQL不支持序列，会创建 一个与序列同名的表代替序列）方式生成ID。

4.4数据库类型切换

在项目中，默认使用数据库自增长（orm.xml）的方式生成ID。ncs-common模块src/main/resource/jpa-orm目录下的orm.xml配置文件 覆盖了实体类中的@GeneratedValue注解设置，如需切换到Oracle或DB2等支持序列的数据库类型时，修改默认值即可（hibernate默认到 META-INF目录下查找orm.xml文件，当profile设置为sequence时，打包不会将orm.xml文件映射到META-INF目录下）。 在mysql配置orm为jpa-orm，而oracle和db2配置为none。 项目pom.xml配置文件:

```
<profile> <id>mysql</id> <activation> <activeByDefault>true</activeByDefault> </activation> <properties> <orm>jpa-orm</orm> <database>mysql</database> </properties> </profile> <profile> <id>oracle</id> <properties> <orm>none</orm> <database>oracle</database> </properties> </profile> <profile> <id>db2</id> <properties> <orm>none</orm> <database>db2</database> </properties> </profile>
```

4.5项目打包

1. `mvn clean package -Dmaven.test.skip=true` 不指定参数时，默认值为-Pdev,mysql。
2. `mvn clean package -Dmaven.test.skip=true -Psit,mysql` 使用mysql数据库,内部使用identity方式生成ID。
3. `mvn clean package -Dmaven.test.skip=true -Psit,oracle` 使用oracle数据库,内部使用sequence方式生成ID。
4. `mvn clean package -Dmaven.test.skip=true -Psit,db2` 使用db2数据库,内部使用sequence方式生成ID。

注意： -Pdev,mysql等价于-Penv=dev,database=mysql;如果设置了env的值，则必须同时设置orm的值（orm默认值会失效）。

谢谢观看！