**Public CMS V4.0**

国产金仓、达梦、南通数据库适配手册

作者：Superego

目录

[第一章 金仓、达梦数据库适配 3](#_Toc516136603)

[1. 添加依赖 3](#_Toc516136604)

[2. 修改配置 3](#_Toc516136605)

[注意事项 4](#_Toc516136606)

[3. 迁移数据 4](#_Toc516136607)

[第二章 南大通用数据库适配 5](#_Toc516136608)

[1. 添加依赖 5](#_Toc516136609)

[2. 修改配置 5](#_Toc516136610)

[3. 迁移数据 10](#_Toc516136611)

#  金仓、达梦数据库适配

## 添加依赖

所有数据库的jar都放在部署项目的\webapps\publiccms\WEB-INF\lib这个下面

添加到build path

如图所示：



## 修改配置

在hibernate配置文件中增加schema配置(这里的public是根据数据库定义的模式)

|  |
| --- |
| hibernate.default\_schema=public |

数据库配置文件格式：

 

### 注意事项

（1）、金仓数据库、达梦数据库无需修改数据库方言

（2）、达梦数据库的适配和金仓数据库的适配是一致的

## 迁移数据

1. 金仓数据库工具使用

 通过金仓数据库的迁移工具将其它数据库迁移的数据以及表结构到金仓数据库管理工具中之后，对其所有表结构有自增的表使用identity这个进行手动添加。

如图所示：



# 南大通用数据库适配

## 添加依赖

所有数据库的jar都放在部署项目的\webapps\publiccms\WEB-INF\lib这个下面

添加到build path

## 修改配置

程序里的主键策略

 修改主键策略为：org.hibernate.id.IncrementGenerator

 在\publiccms\src\main\resources\config\hibernate.properties配置里进行修改为hibernate.dialect=com.publiccms.common.dialect.GBaseDialect;

如图所示：



GBaseDialect.java 源码

|  |
| --- |
| package com.publiccms.common.dialect;import java.sql.SQLException;import org.hibernate.MappingException;import org.hibernate.dialect.Dialect;import org.hibernate.dialect.function.VarArgsSQLFunction;import org.hibernate.exception.spi.TemplatedViolatedConstraintNameExtracter;import org.hibernate.exception.spi.ViolatedConstraintNameExtracter;import org.hibernate.internal.util.JdbcExceptionHelper;import org.hibernate.internal.util.StringHelper;import org.hibernate.type.StandardBasicTypes;//Referenced classes of package org.hibernate.dialect:// Dialectpublic class GBaseDialect extends Dialect { public GBaseDialect() { registerColumnType(-5, "int8"); registerColumnType(-2, "byte"); registerColumnType(-7, "smallint"); registerColumnType(1, "char($l)"); registerColumnType(91, "date"); registerColumnType(3, "decimal"); registerColumnType(8, "double"); registerColumnType(6, "float"); registerColumnType(4, "integer"); registerColumnType(-4, "blob"); registerColumnType(-1, "clob"); registerColumnType(2, "decimal"); registerColumnType(7, "smallfloat"); registerColumnType(5, "smallint"); registerColumnType(93, "datetime year to fraction(5)"); registerColumnType(92, "datetime hour to second"); registerColumnType(-6, "smallint"); registerColumnType(-3, "byte"); registerColumnType(12, "varchar($l)"); registerColumnType(12, 255, "varchar($l)"); registerColumnType(12, 32739, "lvarchar($l)"); registerFunction("concat", new VarArgsSQLFunction(StandardBasicTypes.STRING, "(", "||", ")")); } public String getAddColumnString() { return "add"; } public boolean supportsIdentityColumns() { return true; } public String getIdentitySelectString(String table, String column, int type) throws MappingException { return type != -5 ? "select dbinfo('sqlca.sqlerrd1') from systables where tabid=1" : "select dbinfo('serial8') from systables where tabid=1"; } public String getIdentityColumnString(int type) throws MappingException { return type != -5 ? "serial not null" : "serial8 not null"; } public boolean hasDataTypeInIdentityColumn() { return false; } public String getAddForeignKeyConstraintString(String constraintName, String foreignKey[], String referencedTable, String primaryKey[], boolean referencesPrimaryKey) { StringBuffer result = new StringBuffer(30); result.append(" add constraint ").append(" foreign key (").append(StringHelper.join(", ", foreignKey)) .append(") references ").append(referencedTable); if (!referencesPrimaryKey) result.append(" (").append(StringHelper.join(", ", primaryKey)).append(')'); result.append(" constraint ").append(constraintName); return result.toString(); } public String getAddPrimaryKeyConstraintString(String constraintName) { return (new StringBuilder()).append(" add constraint primary key constraint ").append(constraintName).append(" ") .toString(); } public String getCreateSequenceString(String sequenceName) { return (new StringBuilder()).append("create sequence ").append(sequenceName).toString(); } public String getDropSequenceString(String sequenceName) { return (new StringBuilder()).append("drop sequence ").append(sequenceName).toString(); } public String getSequenceNextValString(String sequenceName) { return (new StringBuilder()).append("select ").append(getSelectSequenceNextValString(sequenceName)) .append(" from systables where tabid=1").toString(); } public String getSelectSequenceNextValString(String sequenceName) { return (new StringBuilder()).append(sequenceName).append(".nextval").toString(); } public boolean supportsSequences() { return true; } public boolean supportsLimit() { return true; } public boolean useMaxForLimit() { return true; } public boolean supportsLimitOffset() { return false; } public String getLimitString(String querySelect, int offset, int limit) { if (offset > 0) throw new UnsupportedOperationException("gbasedbt has no offset"); else return (new StringBuffer(querySelect.length() + 8)).append(querySelect) .insert(querySelect.toLowerCase().indexOf("select") + 6, (new StringBuilder()).append(" first ").append(limit).toString()) .toString(); } public boolean supportsVariableLimit() { return false; } public ViolatedConstraintNameExtracter getViolatedConstraintNameExtracter() { return EXTRACTER; } public boolean supportsCurrentTimestampSelection() { return true; } public boolean isCurrentTimestampSelectStringCallable() { return false; } public String getCurrentTimestampSelectString() { return "select distinct current timestamp from gbasedbt.systables"; } private static ViolatedConstraintNameExtracter EXTRACTER = new TemplatedViolatedConstraintNameExtracter() { @Override protected String doExtractConstraintName(SQLException sqle) throws NumberFormatException { String constraintName = null; int errorCode = JdbcExceptionHelper.extractErrorCode(sqle); if (errorCode == -268) constraintName = extractUsingTemplate("Unique constraint (", ") violated.", sqle.getMessage()); else if (errorCode == -691) constraintName = extractUsingTemplate("Missing key in referenced table for referential constraint (", ").", sqle.getMessage()); else if (errorCode == -692) constraintName = extractUsingTemplate("Key value for constraint (", ") is still being referenced.", sqle.getMessage()); if (constraintName != null) { int i = constraintName.indexOf('.'); if (i != -1) constraintName = constraintName.substring(i + 1); } return constraintName; } };} |

数据库配置文件格式:

 

## 迁移数据

（1）、南大通用数据库的迁移工具，迁移途中需要修改相对应的修饰符。如图所示：



1. 、通过南大通用数据库的迁移工具将其它数据库迁移的数据以及表结构到金仓数据库管理工具中之后，对其所有表结构有自增的表使用**SERIAL8**这个进行手动添加。

如图所示：

