|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  | | --- | | 文档密级：公开 | | | | |  |
|  | | | | |
|  | | 优炫云数据库开发接口  V2.0.0.6（软件版本）  使用手册 | |  | | |  |
|  | |
| **文档版本** | 01 |
| **发布日期** | 2017-04-30 |

|  |
| --- |
| 版权所有 © 北京优炫软件股份有限公司 2017。保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  和其他优炫公司商标均为北京优炫软件股份有限公司的商标。  本文档提及的其他所有商标或注册商标，由各自的所有人拥有。  注意  由于产品版本安装或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。 |

|  |  |
| --- | --- |
| 北京优炫软件股份有限公司（总部） | |
| 地址： | 北京市海淀区知春路6号锦秋国际大厦B座702邮编：100191 |
| 网址： | http://www.uxsino.com |
| 电话： | 010-82886998 |
| 传真： | 010-82886338 |
| 服务热线： | 400-650-7837 |

前言

文档目的

本文档介绍了优炫云数据库图形化管理工具（UXDBAdmin）的安装、卸载以及实现的功能，为软件的使用和维护提供必要的信息。

文档对象

* 技术支持工程师
* 维护工程师

修改记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

文档版本 01 （2017-04-30）

第一次正式发布。

目录

[前言 ii](#_Toc484104651)

[1 JDBC 5](#_Toc484104652)

[1.1 安装jdk搭建及配置eclipse环境 5](#_Toc484104653)

[1.2 新建java project 5](#_Toc484104654)

[1.3 导入uxdb的jdbc包 5](#_Toc484104655)

[1.4 uxdb中创建测试表student 6](#_Toc484104656)

[1.5 新建名为ConnUtil和ConnTest的class 7](#_Toc484104657)

[1.6 运行结果 12](#_Toc484104658)

[1.7 使用分析说明 13](#_Toc484104659)

[1.8 安装hibernate tools 13](#_Toc484104660)

[1.9 导入hibernate所有驱动包 13](#_Toc484104661)

[1.10 创建hibernate.cfg.xml 14](#_Toc484104662)

[1.11 连接数据库并生成源代码 16](#_Toc484104663)

[1.12 修改hibernate.cfg.xml 20](#_Toc484104664)

[1.13 创建hibernate的测试类 21](#_Toc484104665)

[1.14 运行结果 24](#_Toc484104666)

[1.15 使用分析说明 24](#_Toc484104667)

[1.16 安装SpringIDE 25](#_Toc484104668)

[1.17 导入spring所有驱动包 25](#_Toc484104669)

[1.18 导入commons驱动包 25](#_Toc484104670)

[1.19 创建Bean类、接口类、接口实现类、beans.xml配置和测试类 26](#_Toc484104671)

[1.20 运行结果 36](#_Toc484104672)

[1.21 使用分析说明 37](#_Toc484104673)

[1.22 注意点 37](#_Toc484104674)

[2 Node.js 38](#_Toc484104675)

[2.1 安装node.js、webstorm和express 38](#_Toc484104676)

[2.2 创建testnodejs工程 38](#_Toc484104677)

[2.3 安装ux模块 39](#_Toc484104678)

[2.4 创建test.js 39](#_Toc484104679)

[2.5 运行结果 44](#_Toc484104680)

[2.6 使用分析说明 44](#_Toc484104681)

[3 ASP.NET 46](#_Toc484104682)

[3.1 安装Visual Studio 2010 46](#_Toc484104683)

[3.2 新建Webuxdb网站 47](#_Toc484104684)

[3.3 添加库类 47](#_Toc484104685)

[3.4 添加UXDB.cs接口类 47](#_Toc484104686)

[3.5 创建测试窗体 56](#_Toc484104687)

[3.6 运行结果 57](#_Toc484104688)

[3.7 使用分析说明 58](#_Toc484104689)

[4 C&C++ 59](#_Toc484104690)

[4.1 开发环境 59](#_Toc484104691)

[4.2 启动服务 59](#_Toc484104692)

[4.3 连接案例 60](#_Toc484104693)

[4.4 运行结果 65](#_Toc484104694)

[4.5 重要说明 65](#_Toc484104695)

# JDBC

## 安装jdk搭建及配置eclipse环境

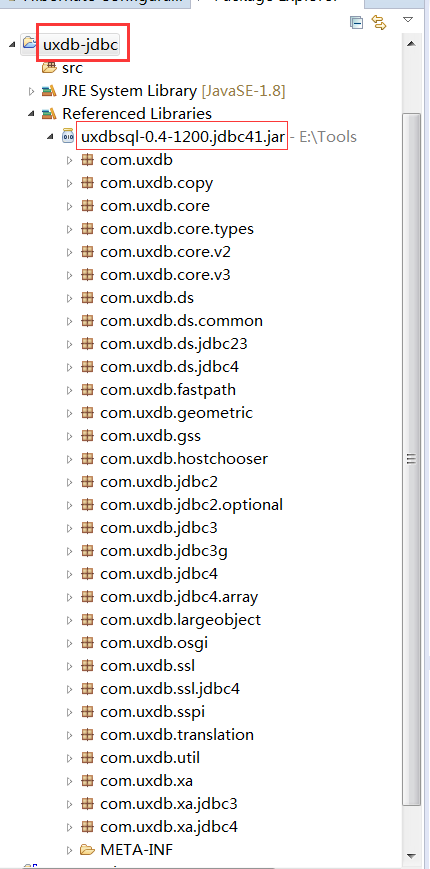
安装jdk7及以上版本。

## 新建java project

打开eclipse，新建名为uxdb-jdbc的java project。

## 导入uxdb的jdbc包

uxdb的jdbc包（uxdbsql-0.4-1200.jdbc41.jar）放在uxdb的安装目录中。



## uxdb中创建测试表student

数据库用户名uxdb、密码123456，创建新的数据库test，在test中创建student表并插入数据，sql语句如下：

create table student(

sno int primary key not null,

sname varchar(20),

sage int,

ssex varchar(6)

);

begin;

insert into student values(1001,'张三',15,'女');

insert into student values(1002,'李四',16,'男');

insert into student values(1003,'王五',15,'男');

insert into student values(1004,'赵六',14,'女');

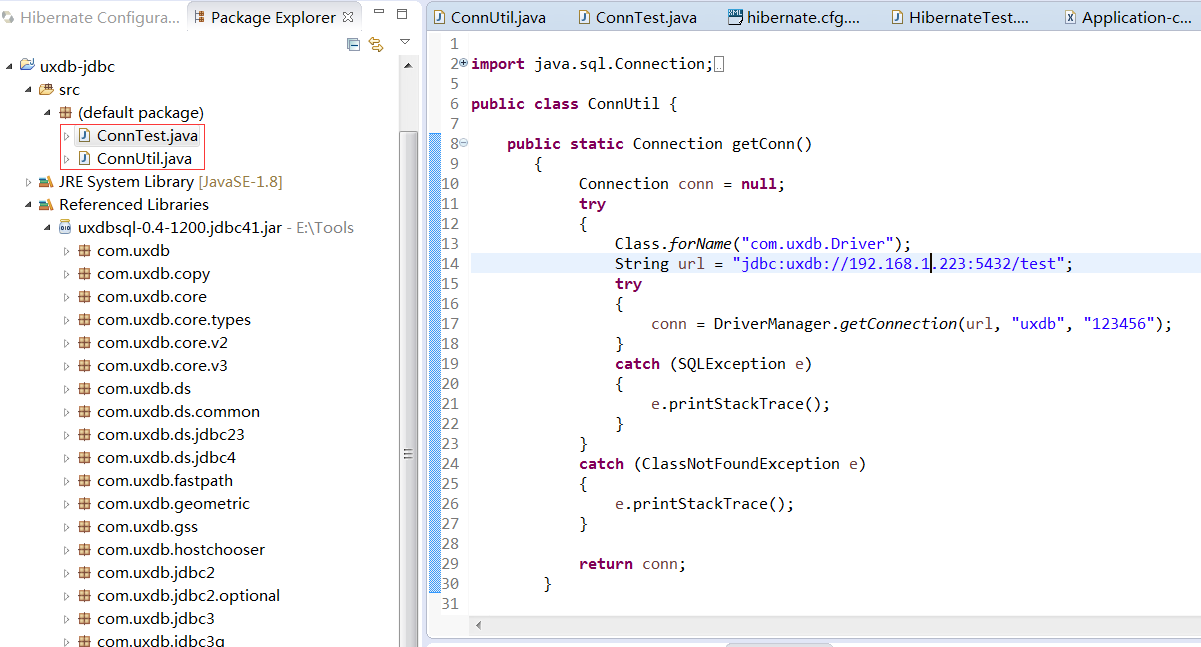
insert into student values(1005,'马七',16,'男');

commit;

## 新建名为ConnUtil和ConnTest的class

uxdb所在机器IP：192.168.1.223

1、ConnUtil.java：



import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class ConnUtil {

public static Connection getConn()

{

Connection conn = null;

try

{

Class.forName("com.uxdb.Driver");

String url = "jdbc:uxdb://192.168.1.223:5432/test";

try

{

conn = DriverManager.getConnection(url, "uxdb", "123456");

}

catch (SQLException e)

{

e.printStackTrace();

}

}

catch (ClassNotFoundException e)

{

e.printStackTrace();

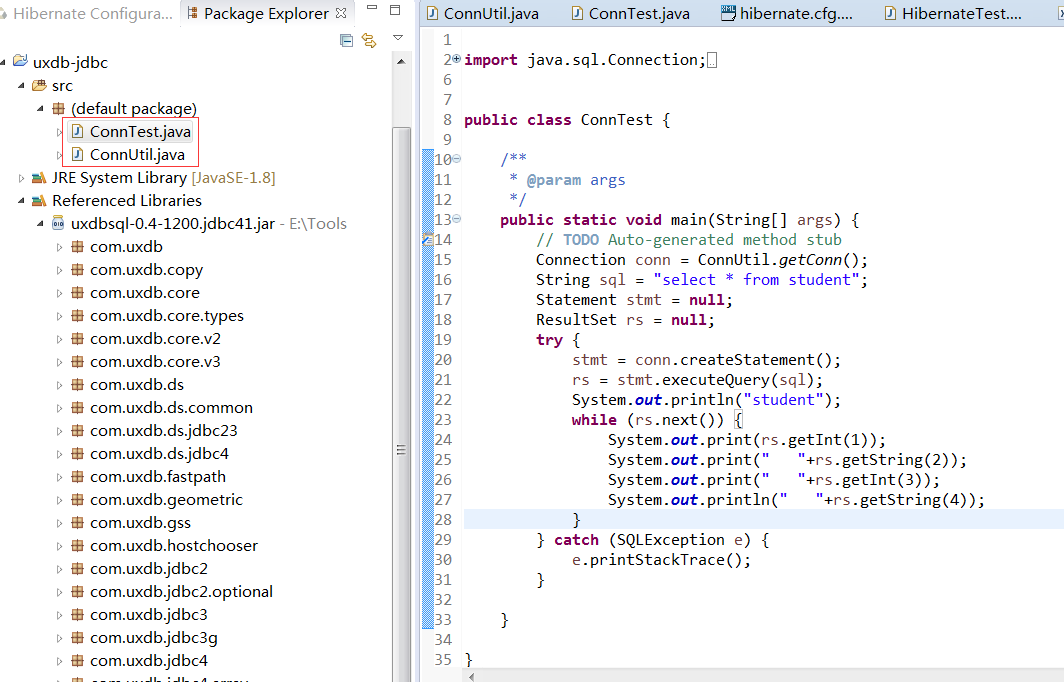
}

return conn;

}

}

2、ConnTest.java：



import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class ConnTest {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

Connection conn = ConnUtil.getConn();

String sql = "select \* from student";

Statement stmt = null;

ResultSet rs = null;

try {

stmt = conn.createStatement();

rs = stmt.executeQuery(sql);

System.out.println("student");

while (rs.next()) {

System.out.print(rs.getInt(1));

System.out.print(" "+rs.getString(2));

System.out.print(" "+rs.getInt(3));

System.out.println(" "+rs.getString(4));

}

} catch (SQLException e) {

e.printStackTrace();

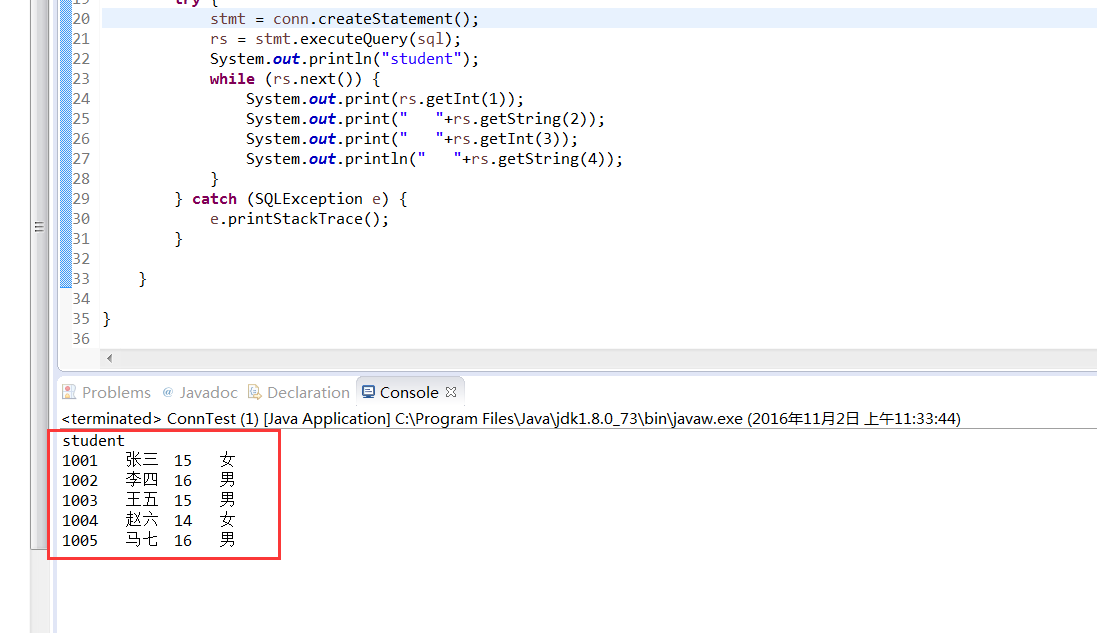
}

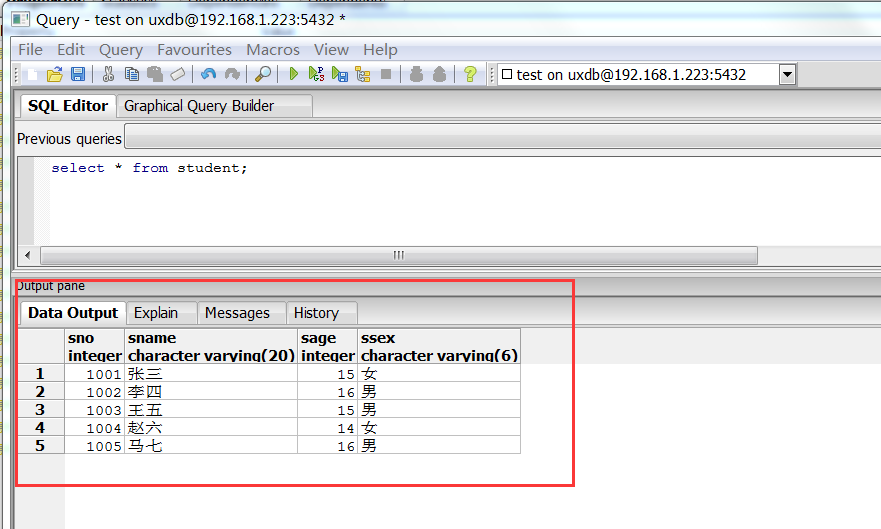
}

}

## 运行结果

运行结果如下显示时jdbc连接使用uxdb成功，显示student表的数据与uxdb数据库中查看结果一致。





## 使用分析说明

1、java.sql.Connection连接数据库类；

2、java.sql.DriverManager驱动管理类；

3、Class.forName("com.uxdb.Driver")调用uxdb的jdbc包中的Driver.class；

4、DriverManager.getConnection("jdbc:uxdb://192.168.1.223:5432/test", "uxdb", "123456")参数格式"jdbc:uxdb://IP:port/databasename", "username", "password"。

## 安装hibernate tools

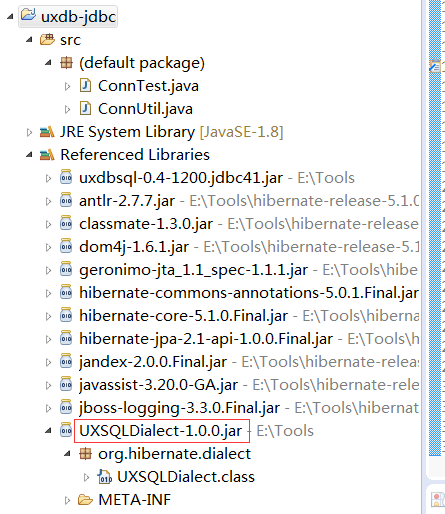
官网下载离线包或者在线安装。

## 导入hibernate所有驱动包

1、导入官网下载的hibernate驱动包；

2、导入uxdb的hibernate驱动包：

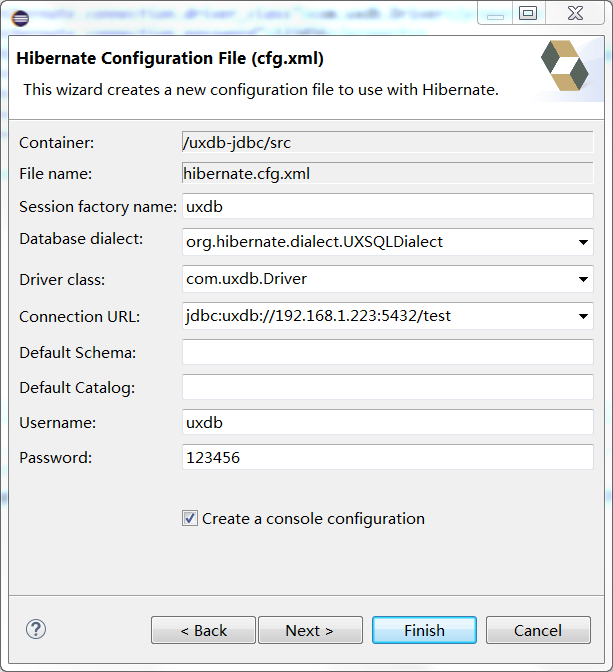
uxdb的hibernate驱动包（UXSQLDialect-1.0.0.jar）在uxdb的安装目录中。



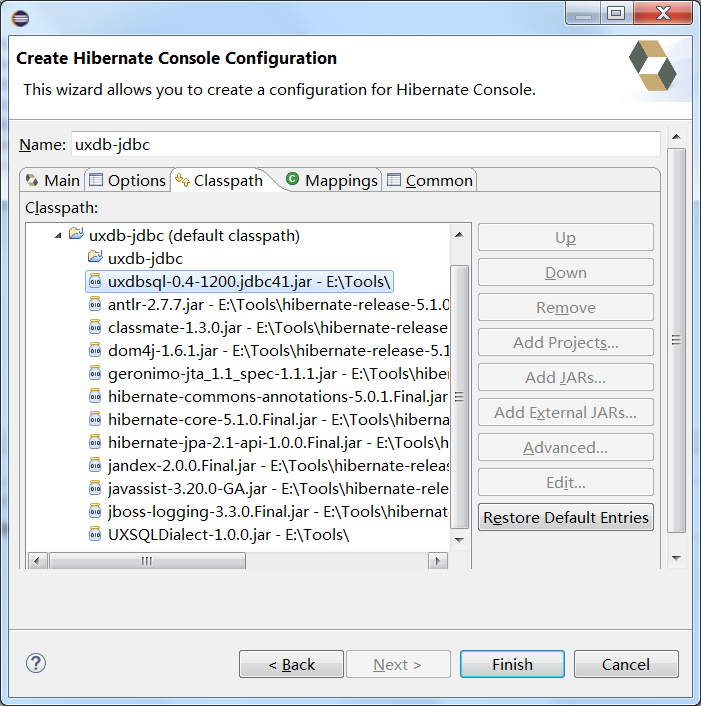
## 创建hibernate.cfg.xml

uxdb server端IP为192.168.1.223，数据库test，用户名uxdb，密码123456

1、创建hibernate.cfg.xml,根据下图填写相关配置；

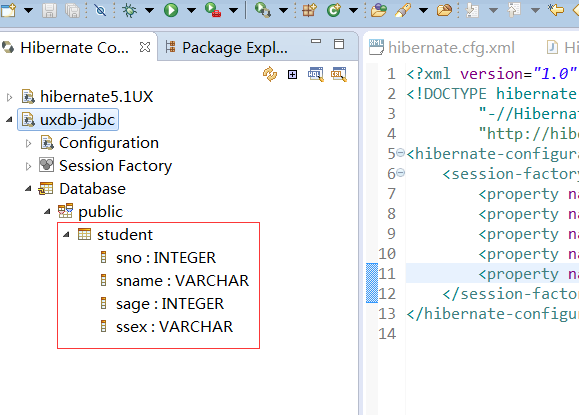


2、Classpath中选择uxdb的JDBC驱动包，然后Finish。



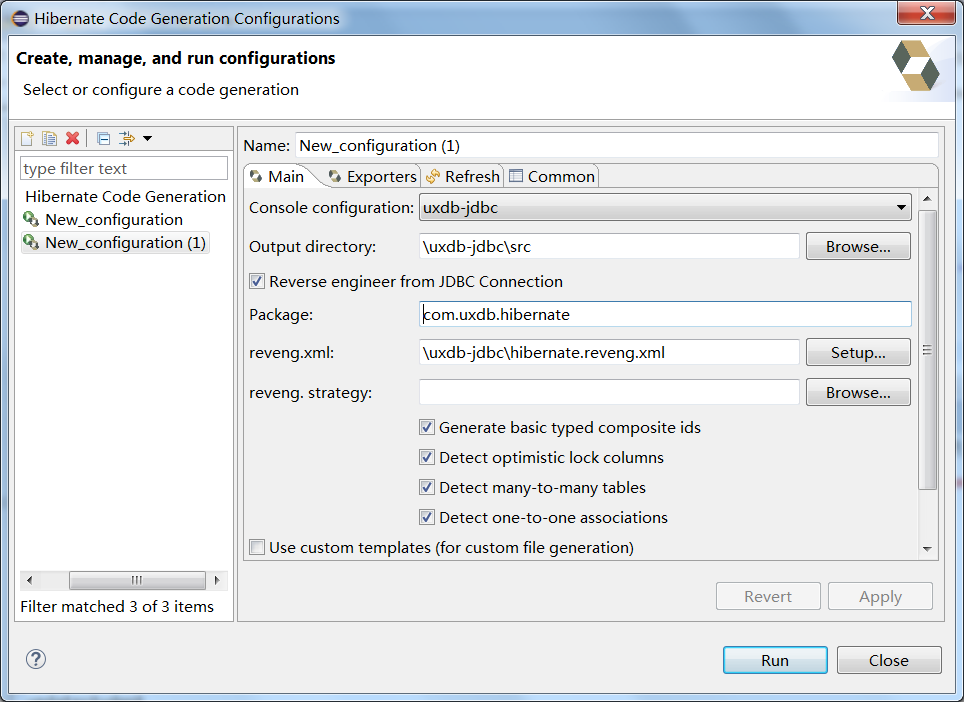
## 连接数据库并生成源代码

1、开Hibernate Configurations的View窗口，如下图显示即数据库连接成功。

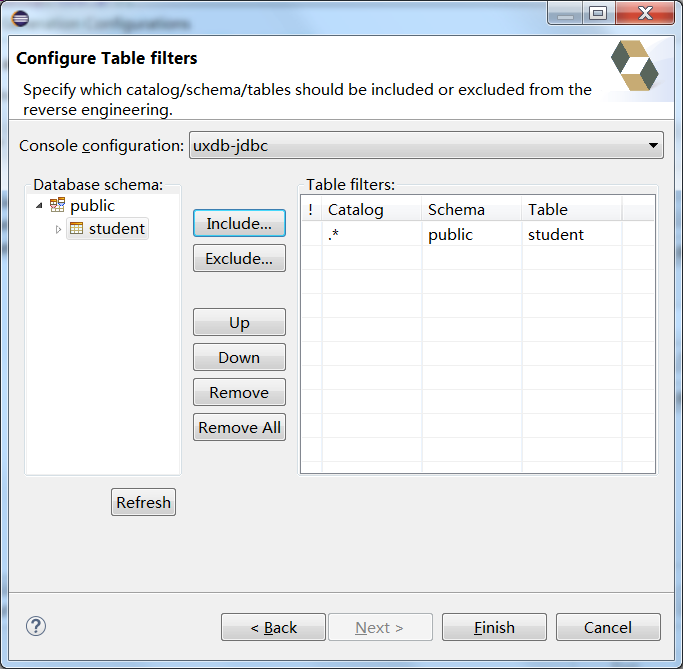


2、对数据库反向生成java代码

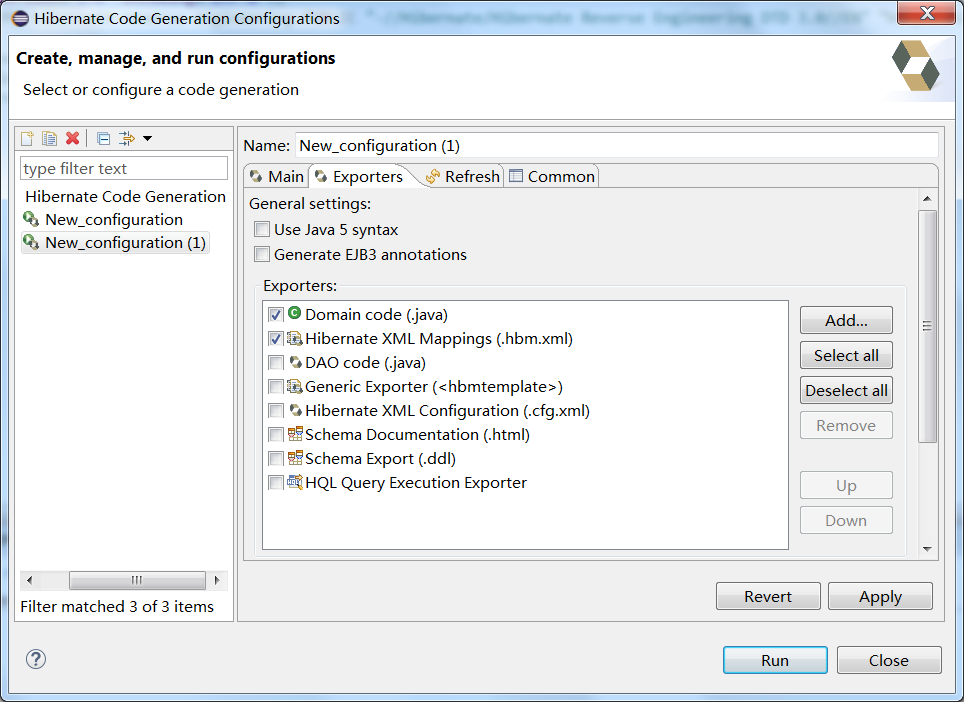
a、新建Hibernate Code generation 的configure；



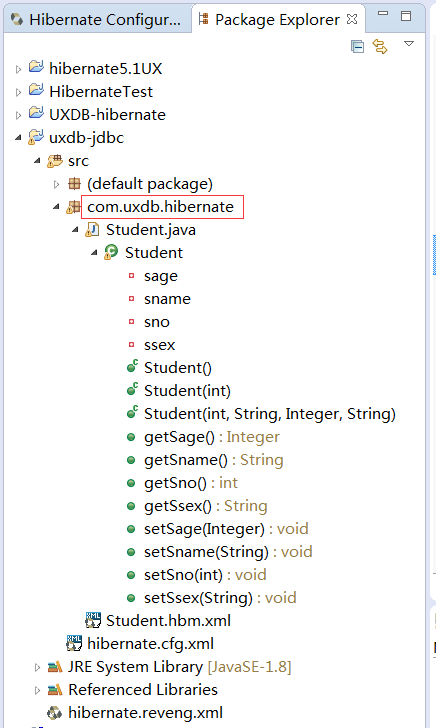
b、点击reveng.xml的Setup…按钮，对数据表进行如下的配置后点击Finish；



c、选择Exporters选项卡，并进行如下的配置，点击Run完成；

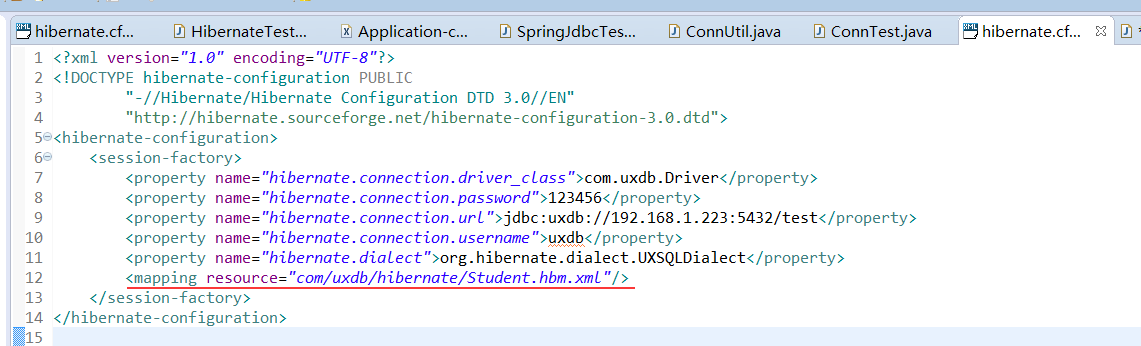


d、如下图在工程中生成了com.uxdb.hibernate包及访问数据库的源代码。



## 修改hibernate.cfg.xml

增加<mapping resource="com/uxdb/hibernate/Student.hbm.xml"/>



## 创建hibernate的测试类

HbiernateTest.java代码如下：

**package** com.uxdb.hibernate;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**public** **class** HibernateTest {

@SuppressWarnings("unchecked")

**public** **static** **void** main(String[] args) {

//读取hibernate.cfg.xml文件

Configuration cfg = **new** Configuration().configure();

//建立SessionFactory

SessionFactory factory = cfg.buildSessionFactory();

Session session = **null**;

**try** {

//取得session

session = factory.openSession();

//开启事务

session.beginTransaction();

Student student = **new** Student();

student.setSno(2001);

student.setSname("testname");

student.setSage(55);

student.setSsex("男");

//保存student对象

session.save(student);

List<Student> list;

String sql = "from Student";

Query query = session.createQuery(sql);

list = (List<Student>)query.list();

System.***out***.println("studentnew");

**for** (Student u : list) {

System.***out***.println(u.getSno() + ";" + u.getSname() + ";" + u.getSage() + ";" + u.getSsex());

}

//提交事务

session.getTransaction().commit();

}**catch**(Exception e) {

e.printStackTrace();

//回滚事务

session.getTransaction().rollback();

}**finally** {

**if** (session != **null**) {

**if** (session.isOpen()) {

//关闭session

session.close();

}

}

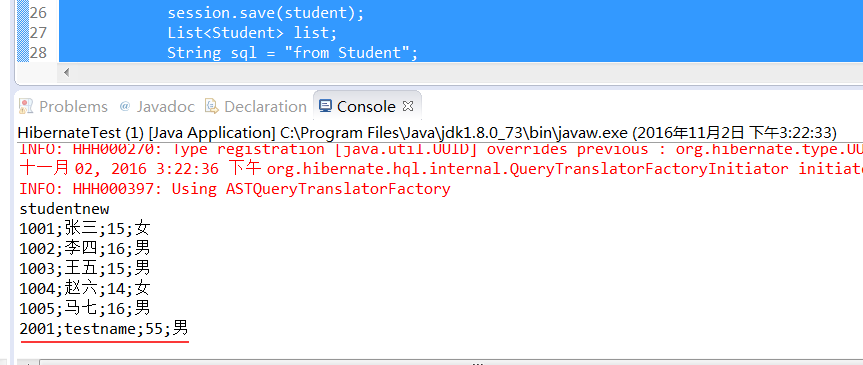
}

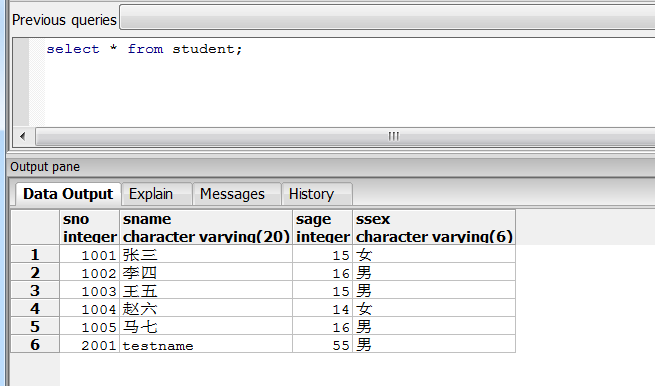
}

}

## 运行结果

运行结果如下显示时hibernate连接使用uxdb成功，显示student表的数据与uxdb数据库中查看结果一致。





## 使用分析说明

创建配置hibernate.cfg.xml时注意以下几项内容：

1、Database dialect：org.hibernate.dialect.UXSQLDialect（使用uxdb的dialect）；

2、Driver class：com.uxdb.Driver（使用uxdb的driver.class）；

3、Connection url：jdbc:uxdb://192.168.1.223:5432/test（jdbc：uxdb：//IP：port/databasename）

4、Username：uxdb

5、Password：123456

## 安装SpringIDE

下载离线包或者在线安装。

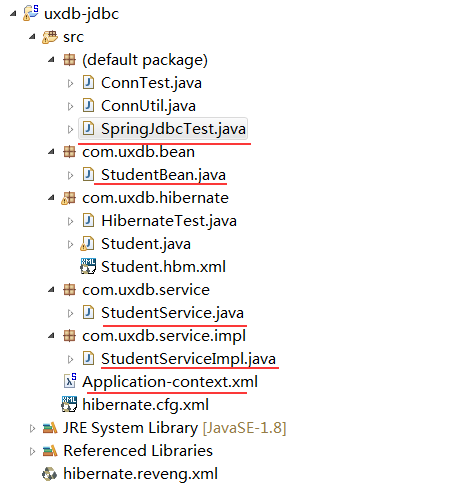
## 导入spring所有驱动包

导入官网下载的spring驱动包。

## 导入commons驱动包

导入官网下载的commons驱动包。

## 创建Bean类、接口类、接口实现类、beans.xml配置和测试类



1、StudentBean.java；

**package** com.uxdb.bean;

**public** **class** StudentBean {

**private** **int** sno;

**private** String sname;

**private** **int** sage;

**private** String ssex;

**public** **int** getSno() {

**return** sno;

}

**public** **void** setSno(**int** sno) {

**this**.sno = sno;

}

**public** String getSname() {

**return** sname;

}

**public** **void** setSname(String sname) {

**this**.sname = sname;

}

**public** **int** getSage() {

**return** sage;

}

**public** **void** setSage(**int** sage) {

**this**.sage = sage;

}

**public** String getSsex() {

**return** ssex;

}

**public** **void** setSsex(String ssex) {

**this**.ssex = ssex;

}

}

2、StudentService.java；

**package** com.uxdb.service;

**import** java.util.List;

**import** com.uxdb.bean.StudentBean;

**public** **interface** StudentService {

//保存

**public** **void** save(StudentBean student);

//更新

**public** **void** update(StudentBean student);

//获取

**public** StudentBean getStudent(**int** sno);

**public** List<StudentBean> getStudentBean();

//删除记录

**public** **void** delete(**int** sno);

}

3、StudentServiceImpl.java；

**package** com.uxdb.service.impl;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.List;

**import** javax.sql.DataSource;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.jdbc.core.RowMapper;

**import** com.uxdb.bean.StudentBean;

**import** com.uxdb.service.StudentService;

**public** **class** StudentServiceImpl **implements** StudentService {

**private** JdbcTemplate jdbcTemplate;

**public** **void** setDataSource(DataSource dataSource) {

**this**.jdbcTemplate = **new** JdbcTemplate(dataSource);

}

@Override

**public** **void** save(StudentBean student) {

// **TODO** Auto-generated method stub

jdbcTemplate.update("insert into student(sno,sname,sage,ssex) values(?,?,?,?)",

**new** Object[] { student.getSno(), student.getSname(), student.getSage(), student.getSsex() },

**new** **int**[] { java.sql.Types.***INTEGER***,java.sql.Types.***VARCHAR***,java.sql.Types.***INTEGER***,java.sql.Types.***VARCHAR*** });

}

@Override

**public** **void** update(StudentBean student) {

// **TODO** Auto-generated method stub

jdbcTemplate.update("update student set sname=?,ssex=? where sno=?",

**new** Object[] { student.getSname(), student.getSsex(), student.getSno() },

**new** **int**[] { java.sql.Types.***VARCHAR***,java.sql.Types.***VARCHAR***,java.sql.Types.***INTEGER*** });

}

@SuppressWarnings("unchecked")

@Override

**public** StudentBean getStudent(**int** sno) {

// **TODO** Auto-generated method stub

**return** (StudentBean) jdbcTemplate.queryForObject("select \* from student where sno=?",

**new** Object[] { sno },

**new** **int**[]{java.sql.Types.***INTEGER***},**new** StudentRowMapper() );

}

@SuppressWarnings("unchecked")

@Override

**public** List<StudentBean> getStudentBean() {

// **TODO** Auto-generated method stub

**return** (List<StudentBean>)jdbcTemplate.query("select \* from student",

**new** StudentRowMapper() );

}

@SuppressWarnings("rawtypes")

**public** **class** StudentRowMapper **implements** RowMapper {

@Override

**public** Object mapRow(ResultSet rs, **int** n) **throws** SQLException {

// **TODO** Auto-generated method stub

StudentBean student=**new** StudentBean();

student.setSno(rs.getInt("sno"));

student.setSname(rs.getString("sname"));

student.setSage(rs.getInt("sage"));

student.setSsex(rs.getString("ssex"));

**return** student;

}

}

@Override

**public** **void** delete(**int** sno) {

// **TODO** Auto-generated method stub

jdbcTemplate.update("delete from student where sno=?", **new** Object[] { sno },

**new** **int**[] { java.sql.Types.***INTEGER*** });

}

}

4、Application-context.xml；

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"* value=*"com.uxdb.Driver"*/>

<property name=*"url"* value=*"jdbc:uxdb://192.168.1.223:5432/test"*/>

<property name=*"username"* value=*"uxdb"*/>

<property name=*"password"* value=*"123456"*/>

</bean>

<bean id=*"txManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

<property name=*"dataSource"* ref=*"dataSource"*/>

</bean>

<bean id=*"studentService"* class=*"com.uxdb.service.impl.StudentServiceImpl"*>

<property name=*"dataSource"* ref=*"dataSource"*></property>

</bean>

</beans>

5、SpringJdbcTest.java；

**import** java.util.List;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.uxdb.bean.StudentBean;

**import** com.uxdb.service.StudentService;

**public** **class** SpringJdbcTest {

**private** **static** StudentService *studentService*;

**private** **static** ApplicationContext *act*;

**public** **static** **void** main(String[] args) {

*act* = **new** ClassPathXmlApplicationContext("Application-context.xml");

*studentService*=(StudentService) *act*.getBean("studentService");

List<StudentBean> list;

StudentBean student = **new** StudentBean();

student.setSno(8008);

student.setSname("springname");

student.setSage(66);

student.setSsex("女");

*studentService*.save(student);

list = *studentService*.getStudentBean();

System.***out***.println("newstudent");

**for** (StudentBean u : list) {

System.***out***.println(u.getSno() + ";" + u.getSname() + ";" + u.getSage() + ";" + u.getSsex());

}

student.setSno(1005);

student.setSname("tnm");

student.setSsex("男");

*studentService*.update(student);

student = *studentService*.getStudent(1005);

System.***out***.println("");

System.***out***.println("updatestudent");

System.***out***.println(student.getSno() + ";" + student.getSname() + ";" + student.getSage() + ";" + student.getSsex());

//studentService.delete(8008);

list = *studentService*.getStudentBean();

System.***out***.println("");

System.***out***.println("student");

**for** (StudentBean u : list) {

System.***out***.println(u.getSno() + ";" + u.getSname() + ";" + u.getSage() + ";" + u.getSsex());

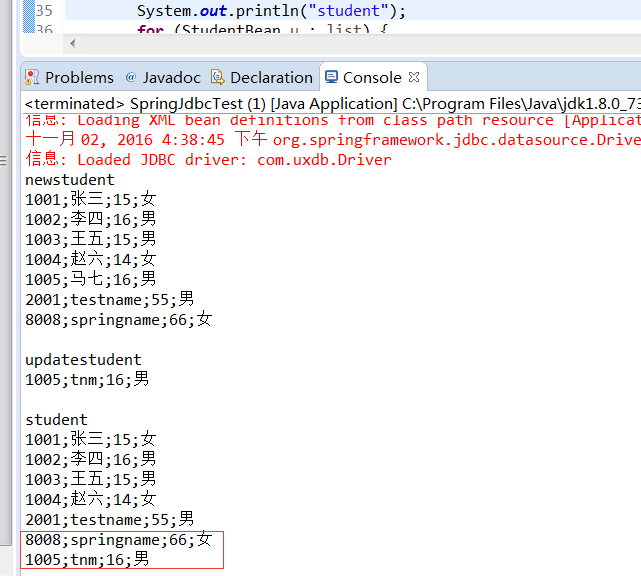
}

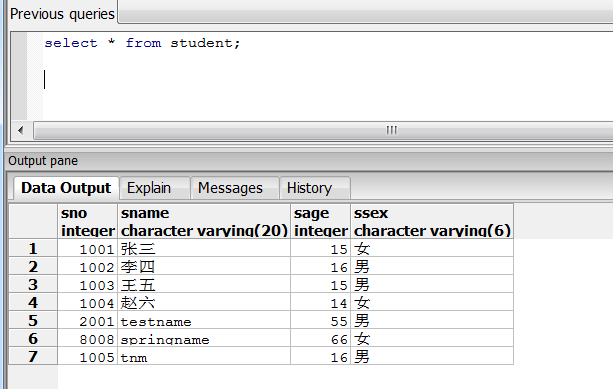
}

}

## 运行结果

运行结果如下显示时spring连接使用uxdb成功，显示student表的数据与uxdb数据库中查看结果一致。





## 使用分析说明

配置Application-context.xml文件注意以下几点：

1、driverClassName="com.uxdb.Driver" （使用uxdb的driver.class）

2、url="jdbc:uxdb://192.168.1.223:5432/test" （jdbc：uxdb：//IP：port/databasename）

3、username="uxdb"

4、password="123456"

## 注意点

1、导入uxdb的jdbc包；

2、使用hibernate框架时在导入uxdb的jdbc包之后需导入hibernate的包和uxdb的Dialect包，并在使用时填写正确。

# Node.js

## 安装node.js、webstorm和express

官网下载并安装node.js、webstorm和express

## 创建testnodejs工程

1、命令执行express -t ejs testnodejs；

2、进入testnodejs文件夹；

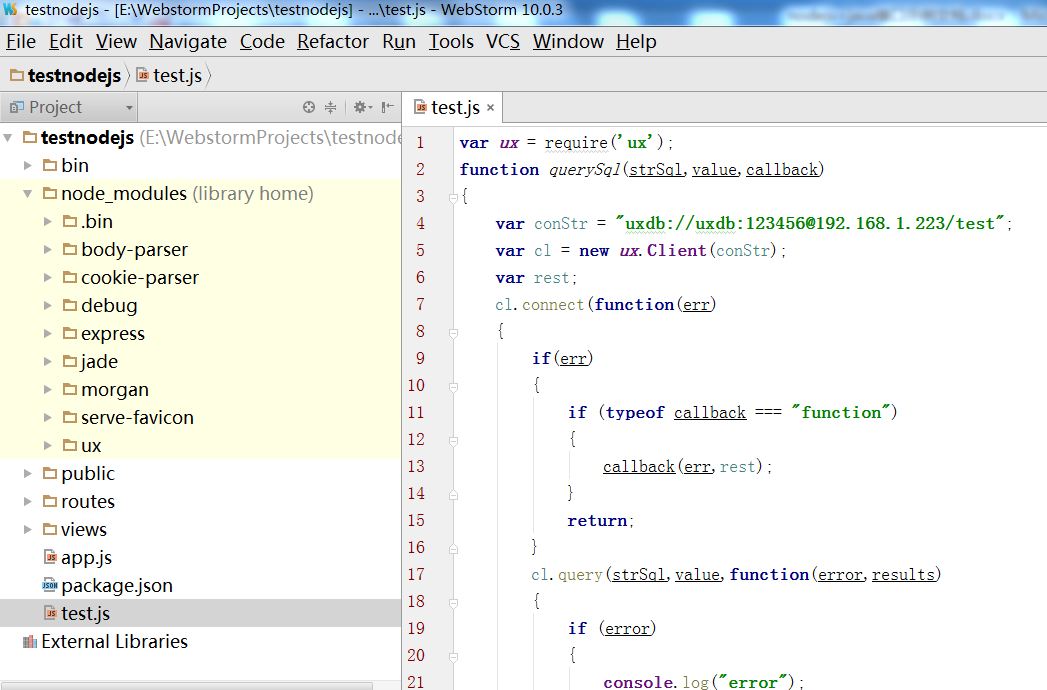
3、执行npm install。

## 安装ux模块

ux模块放在uxdb的安装包目录中。

将安装包目录下uxdb-nodejs中的ux文件夹拷贝到testnodejs工程下的node\_modules文件夹中即可。

## 创建test.js



test.js

var ux = require('ux');

function querySql(strSql,value,callback)

{

var conStr = "uxdb://uxdb:123456@192.168.1.223/test";

var cl = new ux.Client(conStr);

var rest;

cl.connect(function(err)

{

if(err)

{

if (typeof callback === "function")

{

callback(err,rest);

}

return;

}

cl.query(strSql,value,function(error,results)  
 {

if (error)

{

console.log("error");

cl.end();

if (typeof callback === "function")

{

callback(error,0,rest);

}

return;

}

if (results.rowCount > 0)

{

rest = results.rows;

cl.end();

console.log('student');

console.log(rest);

}

if (typeof callback === "function")

{

console.log("[querySql] rst = %s",JSON.stringify(results.rows));

callback(error,results.rowCount,rest);

}

return;

});

});

}

function execSql(text,value,callback)

{

var conStr = "uxdb://uxdb:123456@192.168.1.223/test";

var cl = new ux.Client(conStr);

cl.connect(function(error)

{

if(error)

{

console.log("[execSql]" + error);

if (typeof callback === "function")

{

callback(error);

}

return;

}

cl.query(text,value,function(error,results)

{

if(error)

{

console.log("[execSql]" + error);

cl.end();

}

if(typeof callback === "function")

{

callback(error);

}

cl.end();

return;

});

})

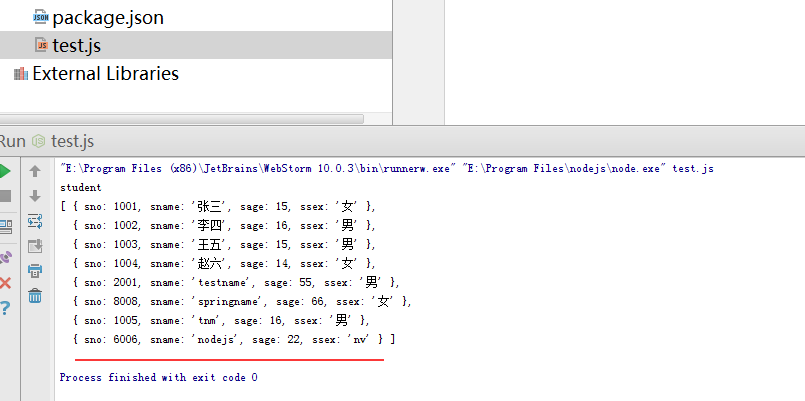
}

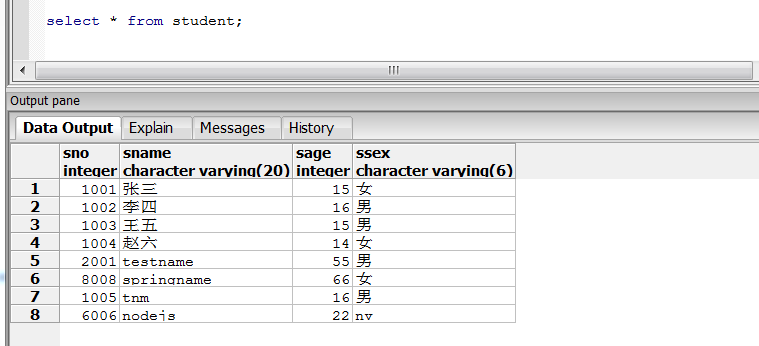
execSql("insert into student values(6006,'nodejs',22,'nv')");

querySql("select \* from student");

## 运行结果

运行结果如下显示时node.js连接使用uxdb成功，显示student表的数据与uxdb数据库中查看结果一致。





## 使用分析说明

1、uxdb端的student表参照JDBC中的第四步进行创建；

2、***ux***.**Client**("")参数格式“**uxdb://username:password@IP/databasename**”，例如代码中的**uxdb://uxdb:123456@192.168.1.223/test**（uxdb-server所在机器IP为192.168.1.223，所连接数据库为test，用户为uxdb，密码为123456）。

# ASP.NET

## 安装Visual Studio 2010

官网下载并安装Visual Studio 2010

## 新建Webuxdb网站

点击文件-->新建-->网站新建webuxdb网站。

## 添加库类

1、添加ASP.NET文件夹-->bin；

2、将Nuxsql.dll和Mono.Security.dll放在bin目录下；

3、添加引用Nuxsql.dll和Mono.Security.dll。

## 添加UXDB.cs接口类

1、添加ASP.NET文件夹-->App\_Code；

2、App\_Code下新建UXDB.cs，代码如下：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//using System.Windows.Forms;

using System.Data;

using Nuxsql;

public class UXDB

{

DataSet DS;

bool ECode;

string ErrString;

NuxsqlConnection Conn = new NuxsqlConnection();

public UXDB(string ServerName, string ServerPort, string DBName, string UserName, string Pwd)

{

ECode = false;

Conn.ConnectionString = "Server=" + ServerName + ";Port=" + ServerPort + ";User Id=" + UserName + ";Password=" + Pwd + ";Database=" + DBName;

try

{

Conn.Open();

}

catch (Exception e)

{

ECode = true;

ErrString = e.Message;

}

}

public DataSet GetRecordSet(string sql)

{

NuxsqlCommand sqlCmd = new NuxsqlCommand();

sqlCmd.Connection = Conn;

sqlCmd.CommandText = sql;

try

{

NuxsqlDataAdapter adp = new NuxsqlDataAdapter(sqlCmd);

DS = new DataSet();

adp.Fill(DS);

}

catch (Exception e)

{

ErrString = e.Message;

ECode = true;

return null;

}

return DS;

}

public int ExecuteSQLScalar(string Sqls)

{

string s;

NuxsqlCommand sqlCmd = new NuxsqlCommand();

sqlCmd.Connection = Conn;

sqlCmd.CommandText = Sqls;

sqlCmd.CommandType = CommandType.Text;

try

{

s = sqlCmd.ExecuteScalar().ToString();

}

catch (Exception e)

{

ErrString = e.Message;

ECode = true;

return -1;

}

return (int.Parse(s));

}

public string ExecuteSQLScalarTOstring(string Sqls)

{

string s;

NuxsqlCommand sqlCmd = new NuxsqlCommand();

sqlCmd.Connection = Conn;

sqlCmd.CommandText = Sqls;

sqlCmd.CommandType = CommandType.Text;

try

{

s = sqlCmd.ExecuteScalar().ToString();

}

catch (Exception e)

{

ErrString = e.Message;

ECode = true;

return "-1";

}

return s;

}

public string ExecuteSQLWithTrans(string Sqls)

{

string s;

NuxsqlTransaction myTrans;

myTrans = Conn.BeginTransaction();

NuxsqlCommand sqlCmd = new NuxsqlCommand();

sqlCmd.Connection = Conn;

sqlCmd.CommandText = Sqls;

sqlCmd.CommandType = CommandType.Text;

sqlCmd.Transaction = myTrans;

sqlCmd.ExecuteNonQuery();

//Sqls="SELECT @@IDENTITY AS ID";

sqlCmd.CommandText = Sqls;

try

{

s = sqlCmd.ExecuteScalar().ToString();

}

catch (Exception e)

{

ErrString = e.Message;

ECode = true;

myTrans.Commit();

return "";

}

myTrans.Commit();

return (s);

}

public void ExecuteSQL(string Sqls)

{

NuxsqlCommand sqlCmd = new NuxsqlCommand();

sqlCmd.Connection = Conn;

sqlCmd.CommandText = Sqls;

sqlCmd.CommandType = CommandType.Text;

try

{

sqlCmd.ExecuteNonQuery();

}

catch (Exception e)

{

ErrString = e.Message;

ECode = true;

}

}

public NuxsqlDataReader DBDataReader(string Sqls)

{

NuxsqlCommand sqlCmd = new NuxsqlCommand();

sqlCmd.Connection = Conn;

sqlCmd.CommandText = Sqls;

sqlCmd.CommandType = CommandType.Text;

try

{

return sqlCmd.ExecuteReader(CommandBehavior.CloseConnection);

}

catch (Exception e)

{

ErrString = e.Message;

ECode = true;

return null;

}

}

public void DBClose()

{

try

{

Conn.Close();

}

catch (Exception e)

{

ErrString = e.Message;

ECode = true;

}

}

public bool ErrorCode()

{

return ECode;

}

public string ErrMessage()

{

return ErrString;

}

~UXDB()

{

}

}

## 创建测试窗体

1、Default.aspx.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using Nuxsql;

public partial class \_Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

//added by qinglong.ou

//NuxDB myDb = new NuxDB("localhost", "5432", "postgres", "postgres", "123456");

UXDB myDb = new UXDB("192.168.1.225", "5432", "uxdb", "uxdb", "123456");

string retStr = "";

string testSql = "select \* from student";

NuxsqlDataReader reader = myDb.DBDataReader(testSql);

// 判断数据是否读到尾.

while (reader.Read())

{

//控制台输入

//string temp = String.Format("{0},{1},{2},{3},{4}", reader[0], reader[1], reader[2], reader[3], reader[4]);

string temp = String.Format("{0},{1},{2},{3}", reader[0], reader[1], reader[2], reader[3]);

//retStr += temp + "<----->";

retStr += temp + "\n";

}

System.Diagnostics.Debug.WriteLine("student\n" + retStr);

Console.WriteLine(retStr);

// 一定要关闭 reader 对象.

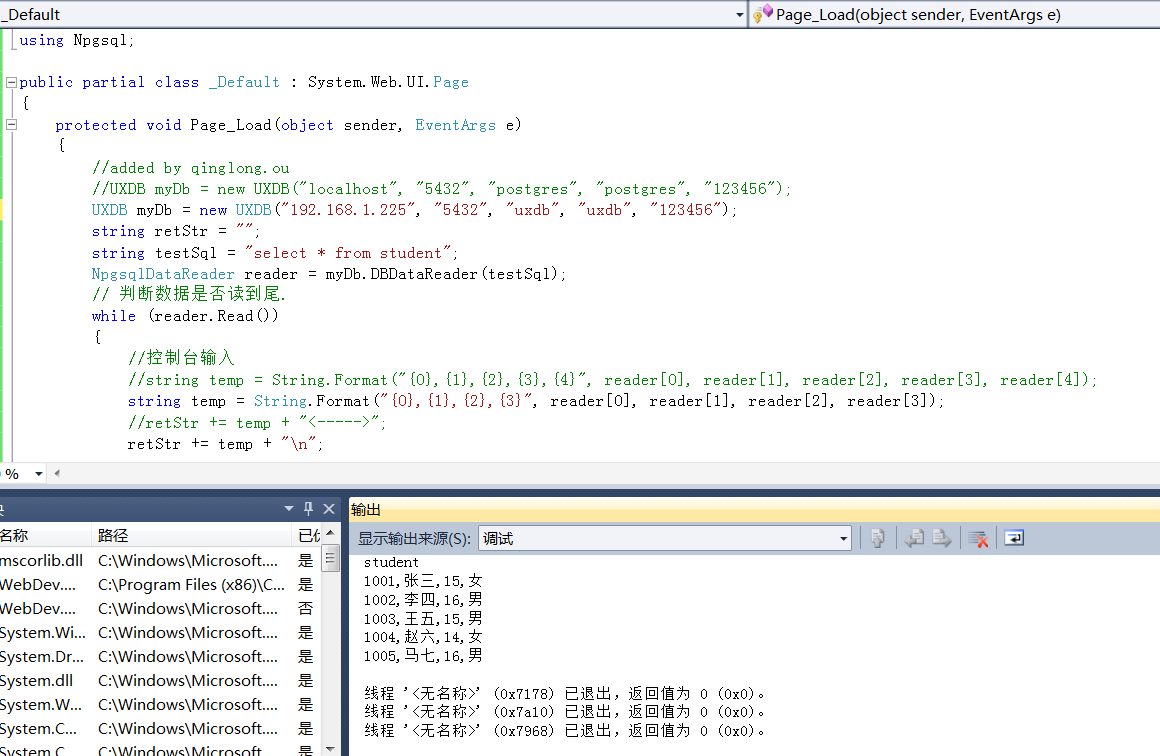
reader.Close();

}

}

## 运行结果

运行结果如下显示时asp.net连接使用uxdb成功，显示student表的数据与uxdb数据库中查看结果一致。



## 使用分析说明

1、uxdb端的student表参照JDBC中的第四步进行创建；

2、**UXDB myDb = new UXDB("IP", "port", "db", "user", "key")**，uxdb-server所在机器IP为192.168.1.225，端口port=5432，所连接db为uxdb，用户user为uxdb，密码为123456。

# C&C++

本文档通过一个简单的案例来介绍C&C++客户端程序通过libuxsql库访问UXDB 实例的方法和步骤。

## 开发环境

这里的开发环境是指C&C++客户端的开发环境， 对数据库服务端的运行环境不做要求。

* OS：Windows7 X64
* tool chain：VS2010 & SP1

## 启动服务

Database server运行在linux环境下， 数据库集群testlocal连接信息：

IP： 192.168.1.122

端口： 5432

用户/口令： uxdb/123456

## 连接案例

* 步骤1 在VS2010中新建“Win32控制台应用程序”， 附加选项选择“空项目”
* 步骤2 主要是项目头文件和链接库文件目录的指定， 根据UXDB服务端的安装目录确定。

简单查询案例：

#include <stdio.h>

#include <stdlib.h>

/\*

\* UXDB客户端头文件

\*/

#include "libuxsql-fe.h"

static void

exit\_nicely(UXconn \*conn)

{

UXSQLfinish(conn);

exit(1);

}

int main(int argc, char \*\*argv)

{

const char \*conninfo;

UXconn \*conn;

UXresult \*res;

int nFields;

int i,

j;

/\*

\* 如果用户在命令行上提供了一个参数，则拿它当作 conninfo 字串使用；

\* 否则缺省为 dbname=uxdb 并且使用环境变量或者所有其它连接参数

\* 都使用缺省值。

\*/

if (argc > 1)

conninfo = argv[1];

else

conninfo = "host=192.168.1.122 port=5432 dbname=uxdb user=uxdb password=123456";

/\* 连接数据库 \*/

conn = UXSQLconnectdb(conninfo);

/\* 检查后端连接成功建立 \*/

if (UXSQLstatus(conn) != CONNECTION\_OK)

{

fprintf(stderr, "Connection to database failed: %s",

UXSQLerrorMessage(conn));

exit\_nicely(conn);

}

/\*

\* 我们的测试实例涉及游标的使用，这个时候我们必须使用事务块。

\* 我们可以把全部事情放在一个 "select \* from ux\_database"

\* UXSQLexec() 里，不过那样太简单了，不是个好例子。

\*/

/\* 开始一个事务块 \*/

res = UXSQLexec(conn, "BEGIN");

if (UXSQLresultStatus(res) != UXRES\_COMMAND\_OK)

{

fprintf(stderr, "BEGIN command failed: %s", UXSQLerrorMessage(conn));

UXSQLclear(res);

exit\_nicely(conn);

}

/\*

\* 应该在结果不需要的时候 UXSQLclear UXresult，以避免内存泄漏

\*/

UXSQLclear(res);

/\*

\* 从系统表 ux\_database（数据库的系统目录）里抓取数据

\*/

res = UXSQLexec(conn, "DECLARE myportal CURSOR FOR select \* from ux\_database");

if (UXSQLresultStatus(res) != UXRES\_COMMAND\_OK)

{

fprintf(stderr, "DECLARE CURSOR failed: %s", UXSQLerrorMessage(conn));

UXSQLclear(res);

exit\_nicely(conn);

}

UXSQLclear(res);

res = UXSQLexec(conn, "FETCH ALL in myportal");

if (UXSQLresultStatus(res) != UXRES\_TUPLES\_OK)

{

fprintf(stderr, "FETCH ALL failed: %s", UXSQLerrorMessage(conn));

UXSQLclear(res);

exit\_nicely(conn);

}

/\* 首先，打印属性名称 \*/

nFields = UXSQLnfields(res);

for (i = 0; i < nFields; i++)

printf("%-15s", UXSQLfname(res, i));

printf("\n\n");

/\* 然后打印行 \*/

for (i = 0; i < UXSQLntuples(res); i++)

{

for (j = 0; j < nFields; j++)

printf("%-15s", UXSQLgetvalue(res, i, j));

printf("\n");

}

UXSQLclear(res);

/\* 关闭入口 ... 我们不用检查错误 ... \*/

res = UXSQLexec(conn, "CLOSE myportal");

UXSQLclear(res);

/\* 结束事务 \*/

res = UXSQLexec(conn, "END");

UXSQLclear(res);

/\* 关闭数据库连接并清理 \*/

UXSQLfinish(conn);

return 0;

}

## 运行结果

datname datdba encoding datcollate datctype datistemplate datallowconn datconnlimit datlastsysoid datfrozenxid datminmxid dattablespace datacl

template1 10 6 en\_US.UTF-8 en\_US.UTF-8 t t -1 13235 1741 1 1663 {=c/uxdb,uxdb=CTc/uxdb}

template0 10 6 en\_US.UTF-8 en\_US.UTF-8 t f -1 13235 1741 1 1663 {=c/uxdb,uxdb=CTc/uxdb}

uxdb 10 6 en\_US.UTF-8 en\_US.UTF-8 f t -1 13235 1741 1 1663

## 重要说明

1、头文件和libuxsql连接依赖于Windows服务端的安装， 主要是include和lib子目录下的头文件和DLL库；

2、连接字符串：host=192.168.1.122 port=5432 dbname=uxdb user=uxdb password=123456