

优炫数据库开发接口使用手册 2.1



UXSINO
优炫软件

优炫数据库开发接口使用手册 2.1

版权 © 2016-2023 北京优炫软件股份有限公司

法律声明

优炫数据库管理系统(简称: UXDB) 是由北京优炫软件股份有限公司开发并发布的一款商业性数据库管理系统。

优炫数据库管理系统(UXDB)的一切知识产权以及与该软件产品相关的所有信息内容,包括但不限于:文字表述及其组合、图标、图饰、图表、色彩、界面设计、版面框架、有关数据、及电子文档等均属北京优炫软件股份有限公司所有。本软件及其文档的任何使用、复制、修改、出租、传播、销售及分发等行为均须经北京优炫软件股份有限公司书面许可。

凡侵犯北京优炫软件股份有限公司知识产权的行为,北京优炫软件股份有限公司将依法追究其法律责任。

本声明的最终解释权归属于北京优炫软件股份有限公司。



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:北京市海淀区学院南路62号中关村资本大厦11层(邮编:100081)
 - 网址: <http://www.uxsino.com>
 - 邮箱: <uxdb_support@uxsino.com>
 - 电话: 010-82886998
 - 传真: 010-82886338
 - 服务热线: 400-650-7837
-

目录

前言	v
1. 文档目的	v
2. 文档对象	v
3. 修改记录	v
1. ASP.NET	1
2. C & C++	13
3. Golang	22
4. JDBC	28
4.1. Hibernate	33
4.2. Spring	46
4.3. MyBatis	55
5. NodeJs	62
6. ODBC	64
7. PHP	70
8. Python	77

表格清单

1. 文档更新记录	v
-----------------	---

前言

1. 文档目的

本文档主要介绍了各种开发接口如何实现连接优炫数据库。

2. 文档对象

- 技术支持工程师
- 维护工程师

3. 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。

表 1. 文档更新记录

工具版本	发布日期	修改说明
2.1.1.5C	2022-12-23	第一次正式发布。

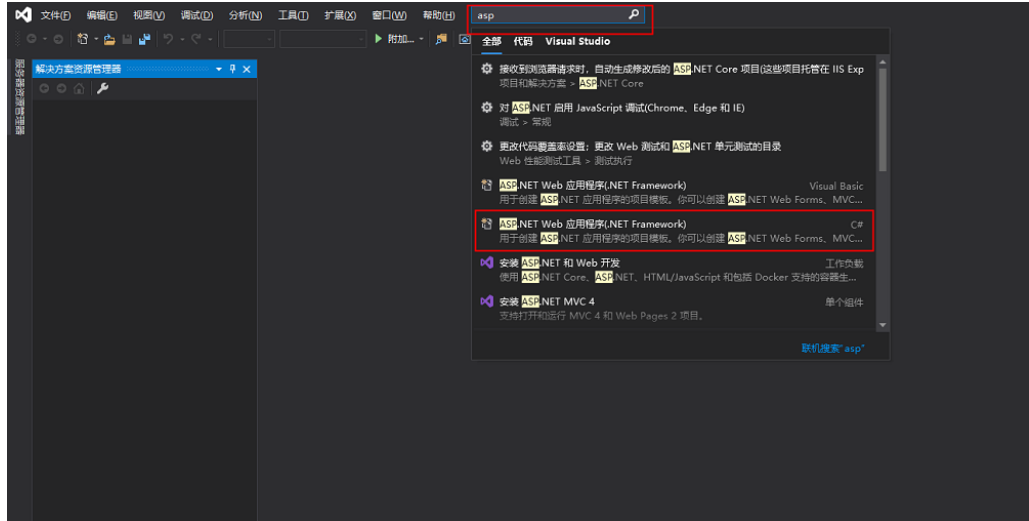
第 1 章 ASP.NET

1. 安装Visual Studio 2019

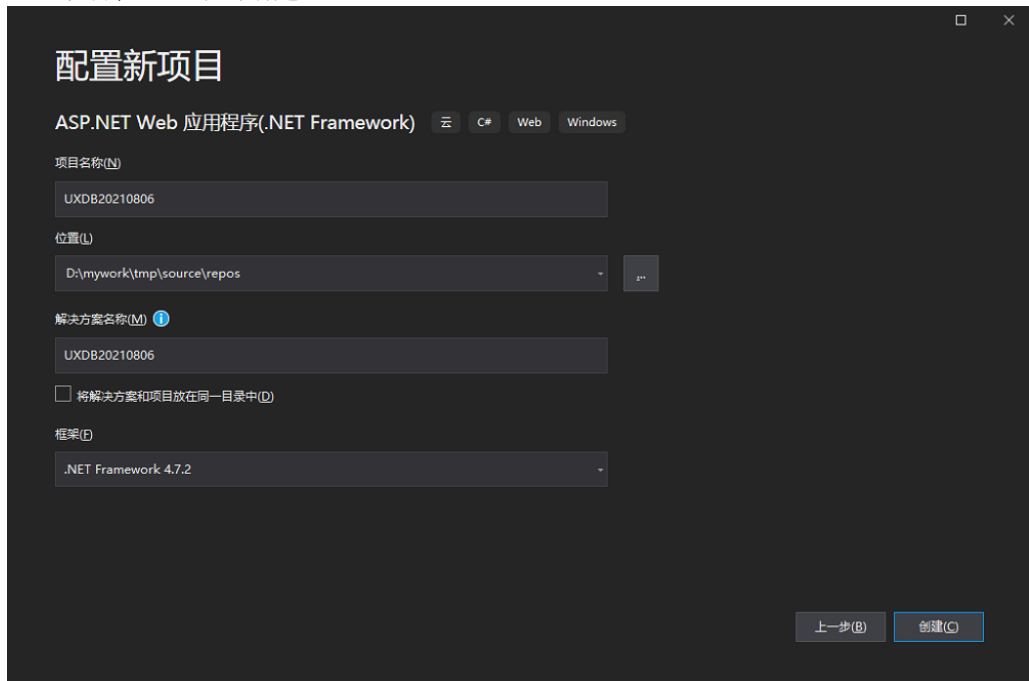
官网下载并安装Visual Studio 2019。

2. 新建Webuxdb网站

a. 打开Visual Studio 2019，在搜索框中输入asp，选择如下选项。



b. 配置项目信息，单击新建。

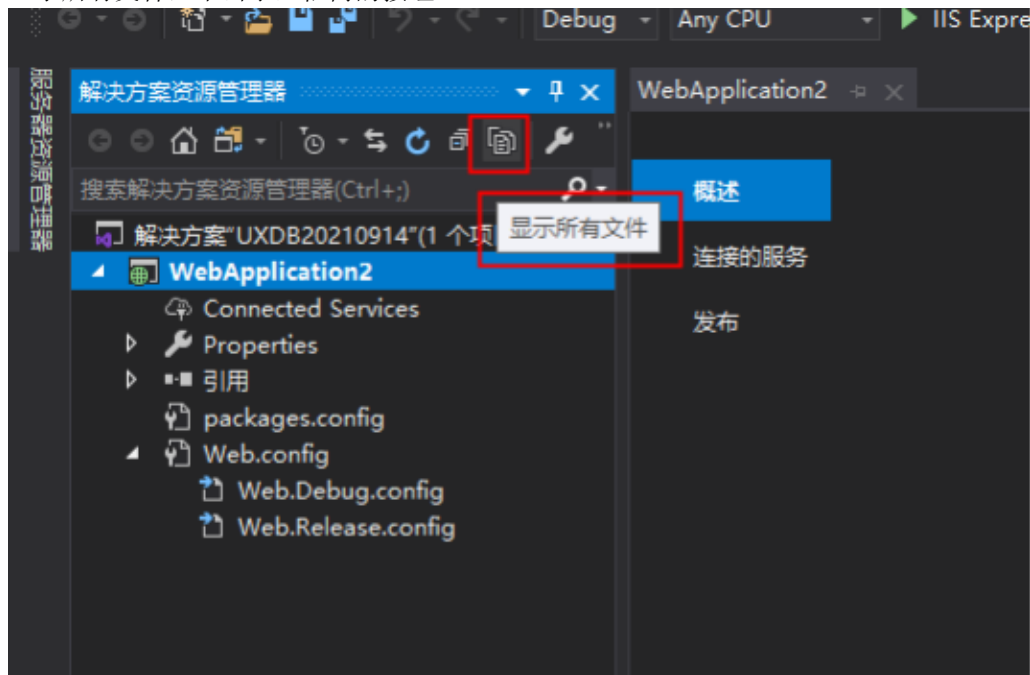


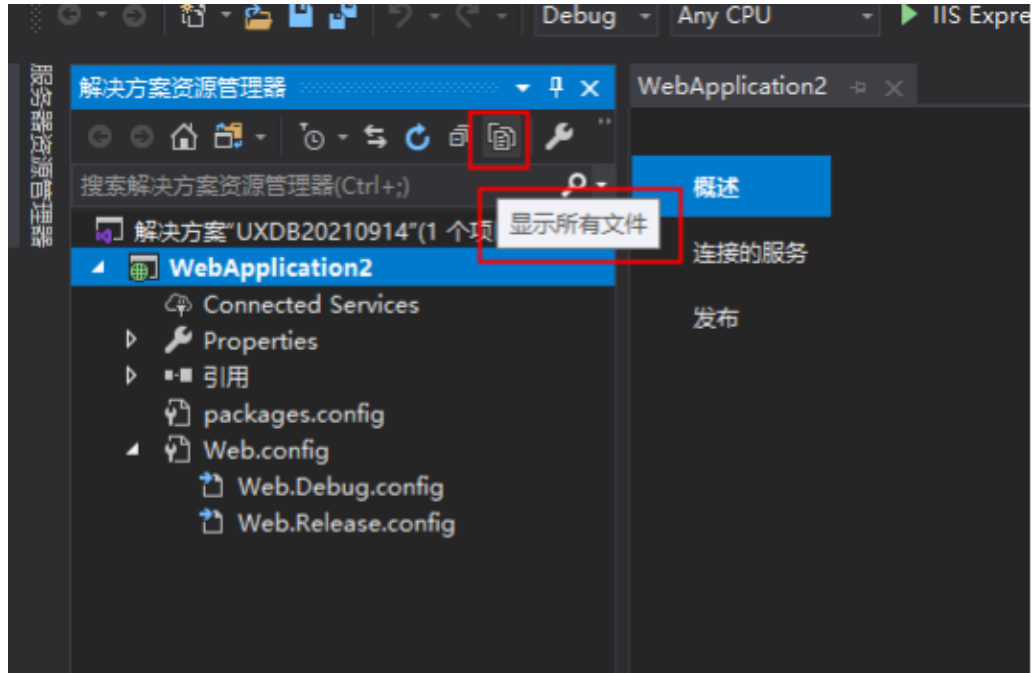
c. 创建空项目。



3. 添加库类

- a. 显示所有文件，单击小红框内的按钮。





- b. 拷贝Nuxsql.dll和Mono.Security.dll

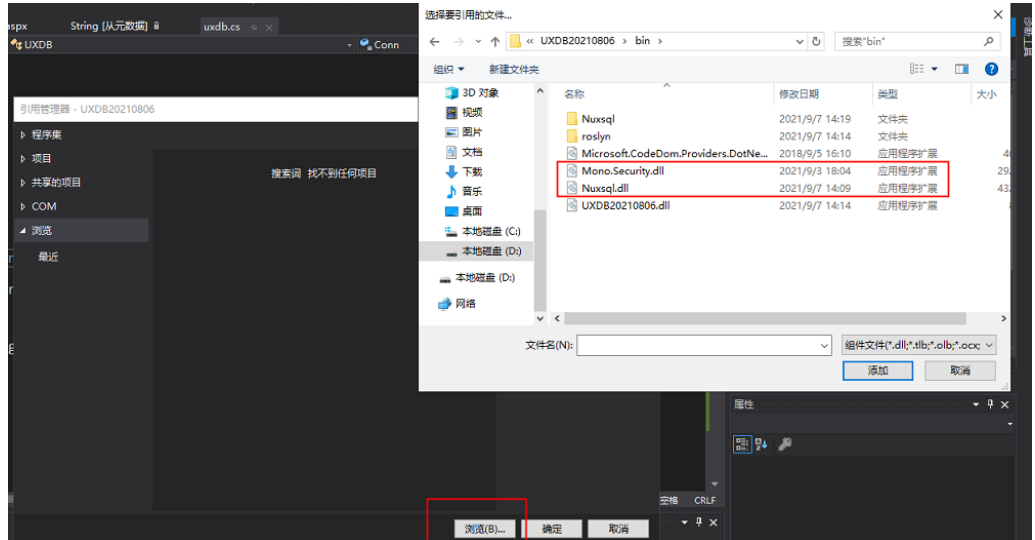
将Nuxsql.dll和Mono.Security.dll拷贝到项目的bin目录文件夹下。

- c. 添加库文件

在项目中，右键“Bin”=>“添加”=>“现有项”，选择Nuxsql.dll和Mono.Security.dll，单击添加。

- d. 添加引用

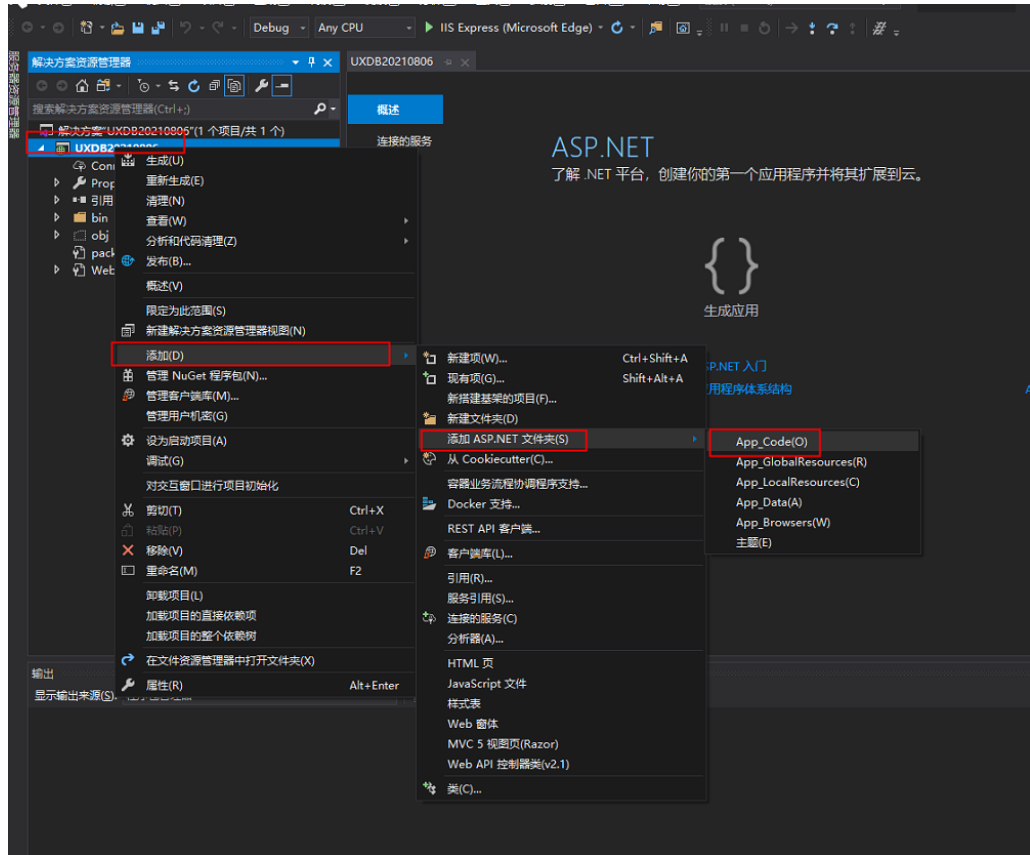
在项目中，右键单击“引用”=>“添加应用”=>浏览选中Nuxsql.dll和Mono.Security.dll，单击确定。



4. 添加UXDB.cs接口类

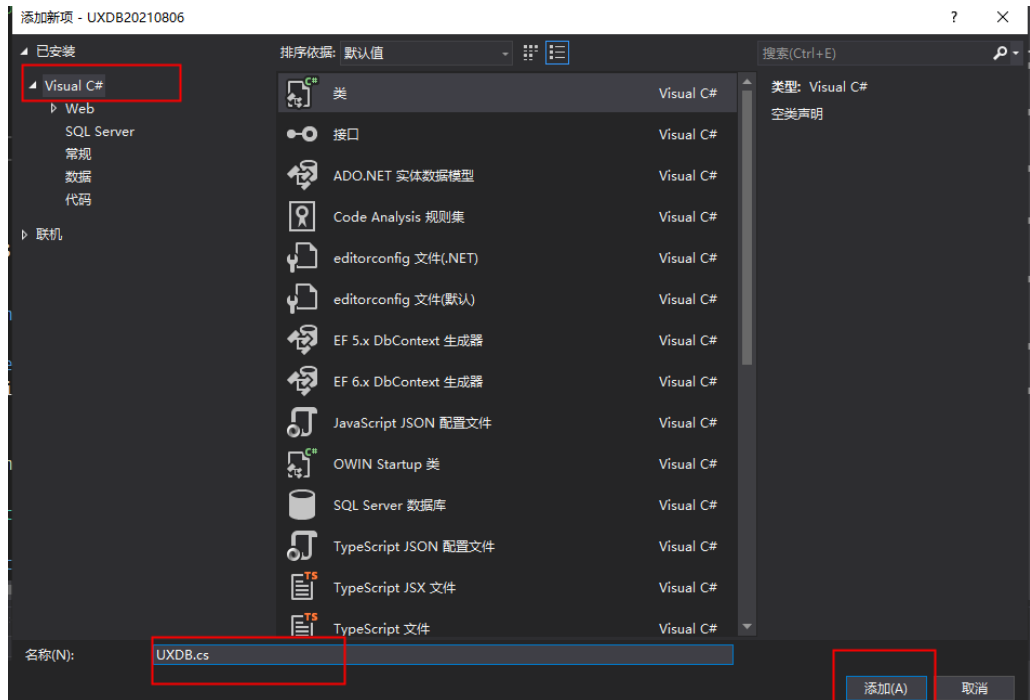
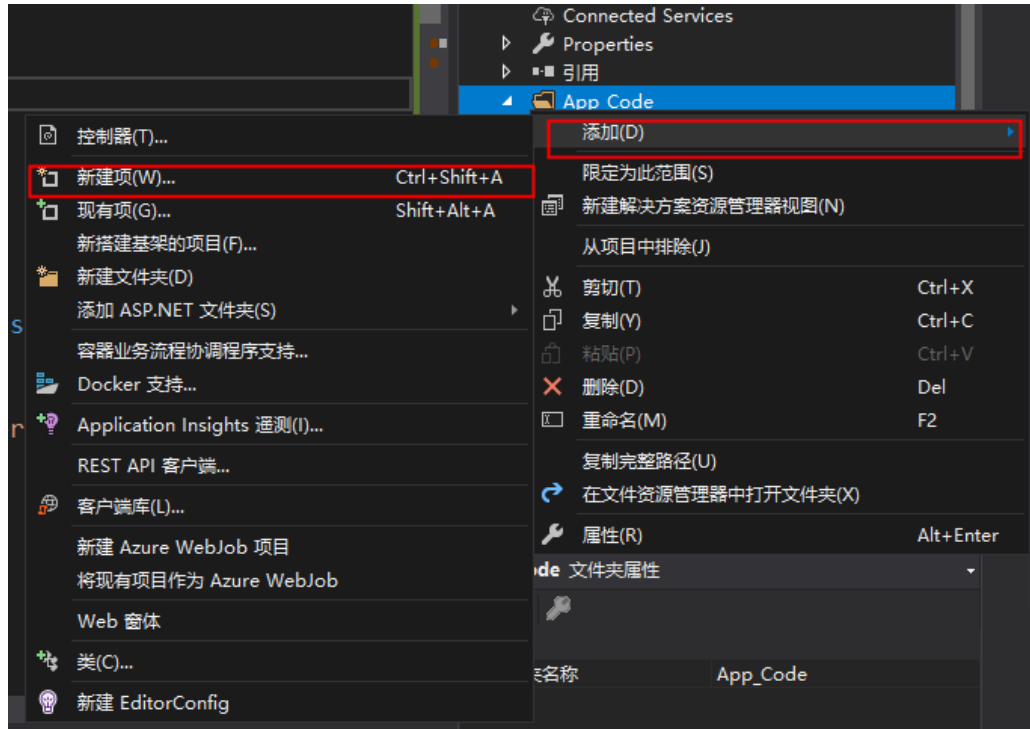
a. 新增App_Code

右键“项目名称”，选择“添加”=>“添加ASP.NET”=>“App_Code”。



b. 新建UXDB.cs

App_Code下新建UXDB.cs，右键“App_Code”=>“添加”=>“添加新项”=>选择Visual C#的类=>修改文件名UXDB.cs，单击添加。



- c. 添加UXDB.cs的代码，如下所示。

```
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()
    {
    }
}

```

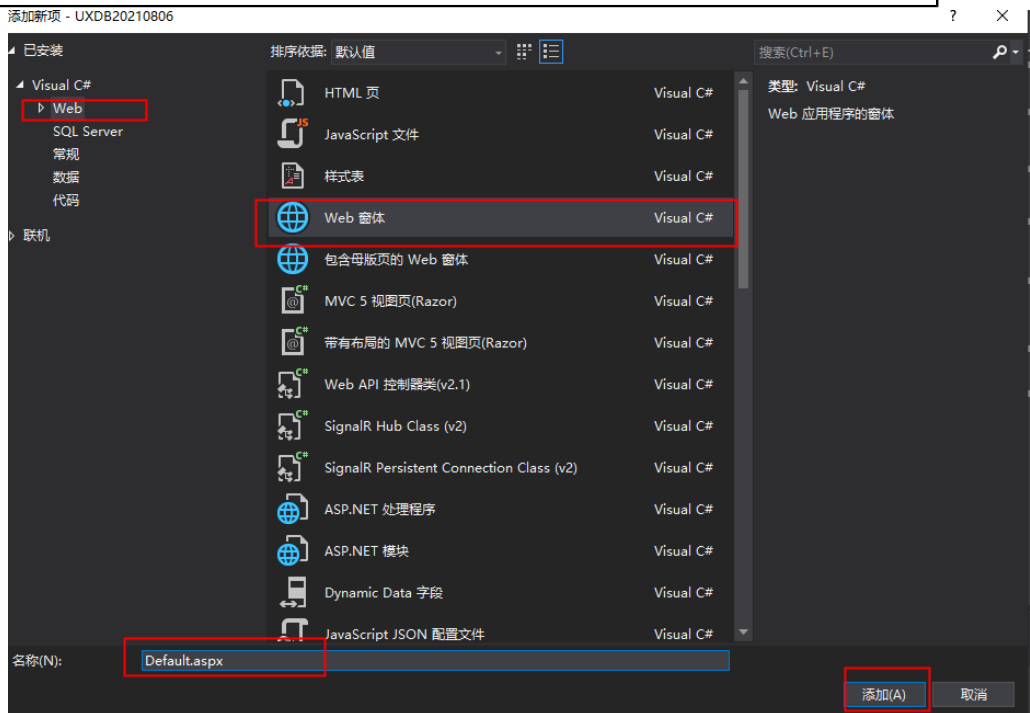
5. 创建窗体

a. 新建Default.aspx

右键“项目名称”，“添加”=>“添加新项”=>“Visual C#”=>“Web窗体”，单击添加。

注意

文件名必须为Default.aspx，在后面的编译过程中类名和文件名必须保持一致。



b. Default.aspx文件中添加测试代码

Default.aspx.cs代码如下所示。

```

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("192.168.1.82", "5432", "test", "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();
}
}
```

6. 运行结果

右键“项目名称” => “生成网站”，启动调试，查看即时窗口输出。

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。

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows a snippet of C# code: `string UserName, string` and `erverPort + ";User Id=" + UserName + ";Password="`. A dropdown menu for `Nuxsql` is visible.
- Diagnostic Tools (诊断工具):** Located on the right, it shows a diagnostic session of 29:36 minutes. It includes a progress bar for the session, a memory usage graph (进程内存) showing 47 MB, and a CPU usage graph (CPU) showing 0%.
- Output Window (输出):** Located at the bottom, it shows the following text:

```
显示输出来源(S): 调试
iisexpress.exe (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132754689534095286): 已加载 "C:\Users\xudb\AppData\Local\Temp\1\Temporary ASP.NET
"iisexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132754689534095286): 已加载 "C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.W
"iisexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132754689534095286): 已加载 "C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.W
student
1001, 张三, 15, 女
1002, 李四, 16, 男
1003, 王五, 15, 男
1004, 赵六, 14, 女
1005, 马七, 16, 男
"iisexpress.exe" (CLR v4.0.30319: DefaultDomain): 已加载 "C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Drawing\v4.0.4.0.0.0_b03
线程 0x4c90 已退出, 返回值为 0 (0x0)。
"iisexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132754689534095286): 已加载 "C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.W
"iisexpress.exe" (CLR v4.0.30319: Domain 2): 已卸载 "C:\Windows\Microsoft.Net\assembly\GAC_32\mscorlib\v4.0.4.0.0.0_b77a5c561934e089'
"iisexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132754689534095286): 已卸载 "C:\Windows\Microsoft.Net\assembly\GAC_32\System.Web
"iisexpress.exe" (CLR v4.0.30319: /LM/W3SVC/2/ROOT-1-132754689534095286): 已卸载 "C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System\ve
```



```

uxdb=# select * from student;
 sno | sname | sage | ssex
-----+-----+-----+-----
 1001 | 张三  |   15 | 女
 1002 | 李四  |   16 | 男
 1003 | 王五  |   15 | 男
 1004 | 赵六  |   14 | 女
 1005 | 马七  |   16 | 男
(5 行记录)

uxdb=# select * from student;
 sno | sname | sage | ssex
-----+-----+-----+-----
 1001 | 张三  |   15 | 女
 1002 | 李四  |   16 | 男
 1003 | 王五  |   15 | 男
 1004 | 赵六  |   14 | 女
 1005 | 马七  |   16 | 男
(5 行记录)

```

7. 使用分析说明

创建uxdb端的student表。

```

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;

```

UXDB myDb = new UXDB("IP", "port", "db", "user", "key"), uxdb-server所在机器IP为192.168.1.82, 端口port为5432, 所连接db为test, 用户user为uxdb, 密码为123456。

第 2 章 C & C++

本章通过一个简单的示例来介绍C & C++客户端程序通过libuxsql库访问UXDB实例的方法和步骤。

1. 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: Windows10 X64

tool chain: VS2010 & SP1

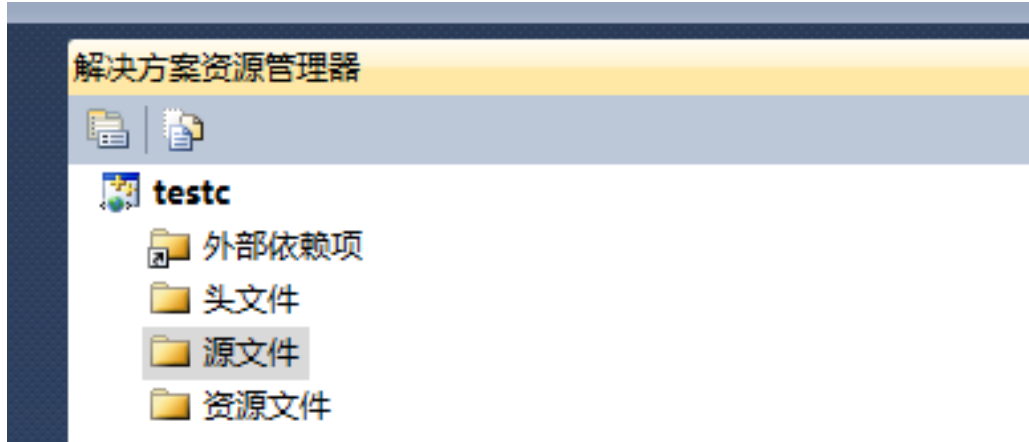
2. 启动服务

Database server运行在linux环境下，数据库集群testlocal连接信息：IP为192.168.1.82，端口为5432，用户名为uxdb，密码为123456。

3. 连接示例

- a. 在VS2010中新建“Win32控制台应用程序”，附加选项选择“空项目”。选择“文件”=>“新建项目”=>“Visual C++”=>“Win32 控制台应用程序”，自定义名称（例：testc），单击“确定”=>“下一步”=>“附加选项中选择空项目”=>单击“完成”。





- b. 右键“源文件” => “添加” => “新建项” => “C++文件(.cpp)”，名称自定义输入（例：main）。代码如下：

```
#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=192.168.1.82 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。
 */

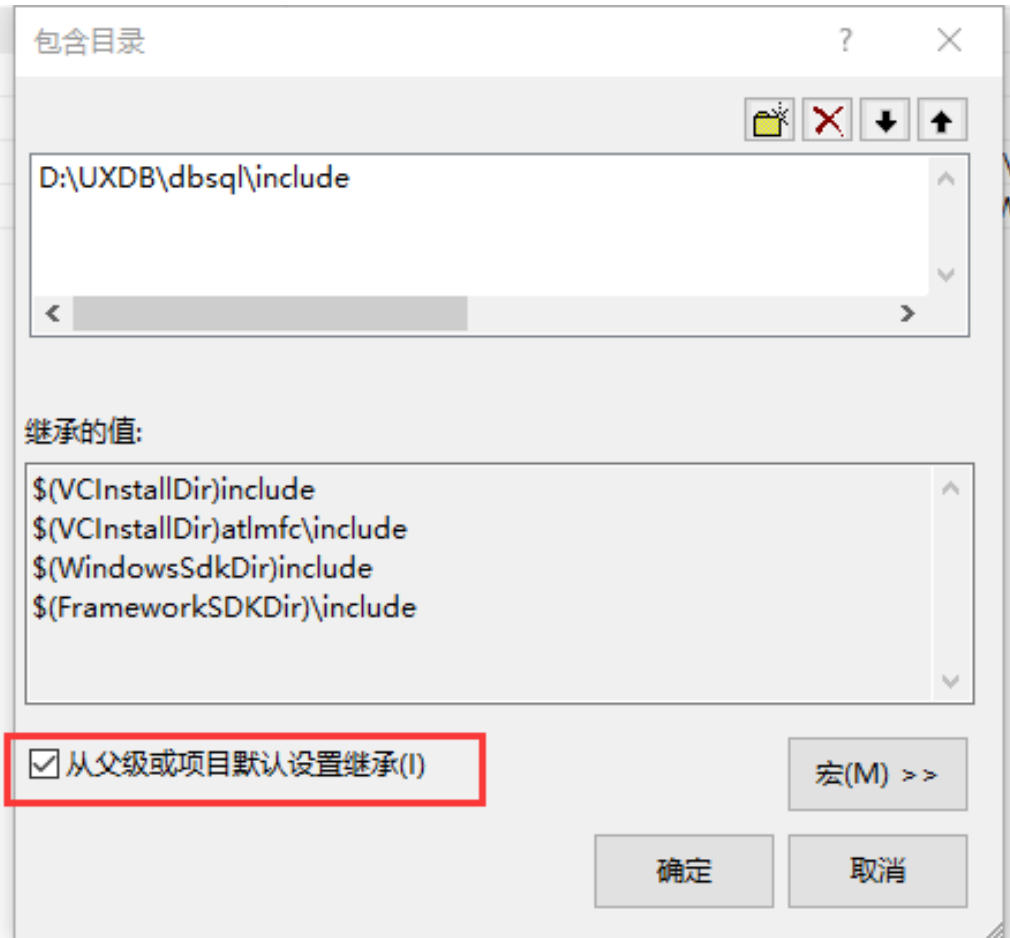
/* 开始一个事务块 */
res = UXSQLexec(conn, "BEGIN");
if (UXSQLresultStatus(res) != UXRES_COMMAND_OK)
{
    fprintf(stderr, "BEGIN command failed: %s", UXSQLErrorMessage(conn));
    UXSQLclear(res);
    exit_nicely(conn);
}
/*
 * 应该在结果不需要的时候 UXSQLclear UXresult，以避免内存泄漏
 */
UXSQLclear(res);
/*
 * 从系统表 ux_database（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");

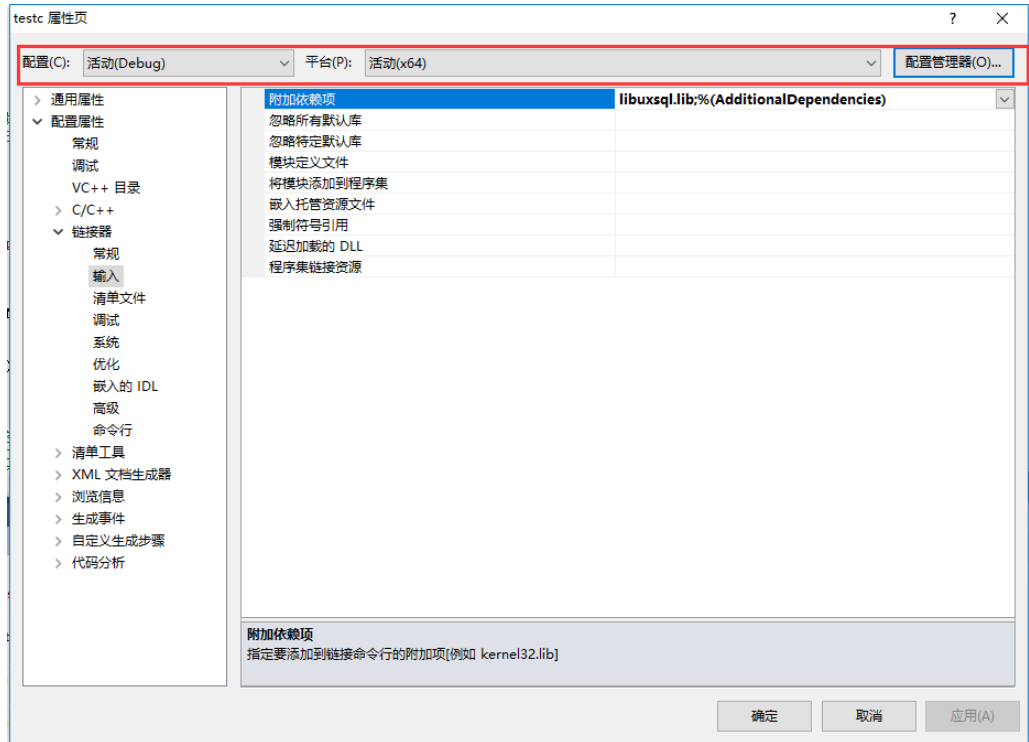
```

```
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;
}
```

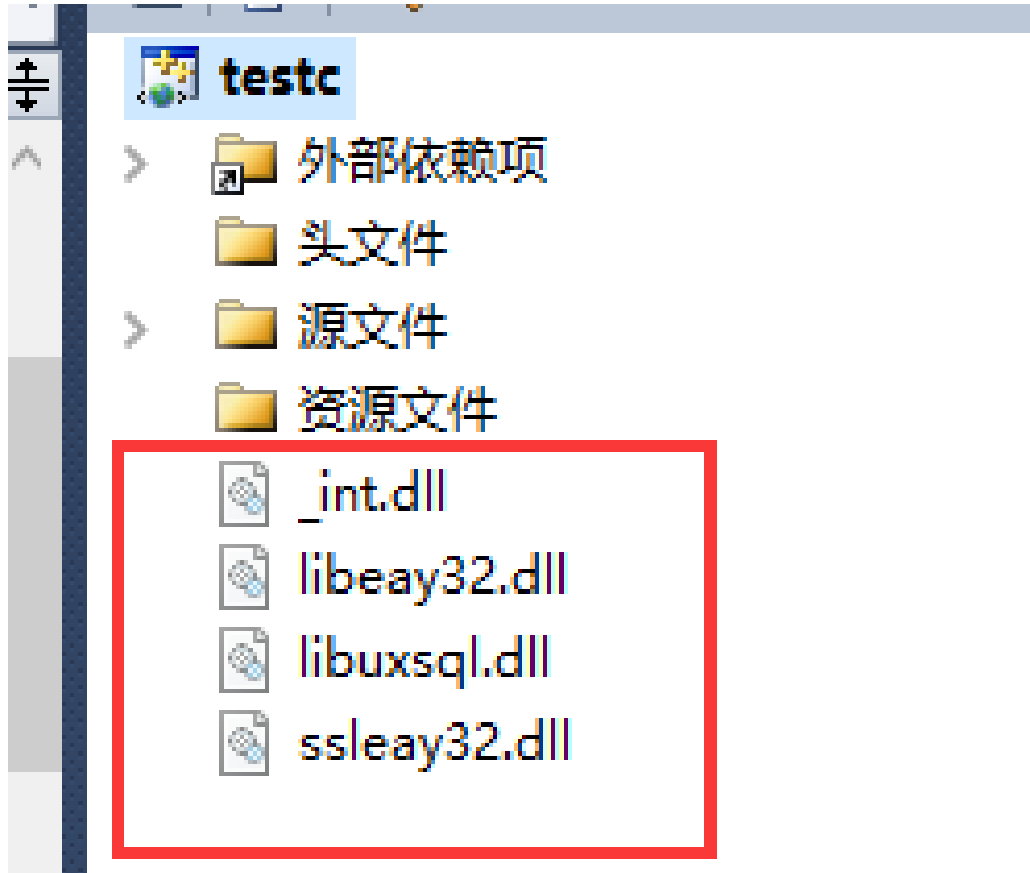
- c. 根据UXDB服务端的安装目录指定项目头文件和链接库文件目录。

右键“项目” => “属性” => “配置属性” => “VC++目录”，分别选择包含目录和库目录，进行编辑，分别添加uxdb\dbsql下的include目录和lib目录（编辑目录时勾选从父级或项目默认设置继承，目录根据实际路径编辑）。

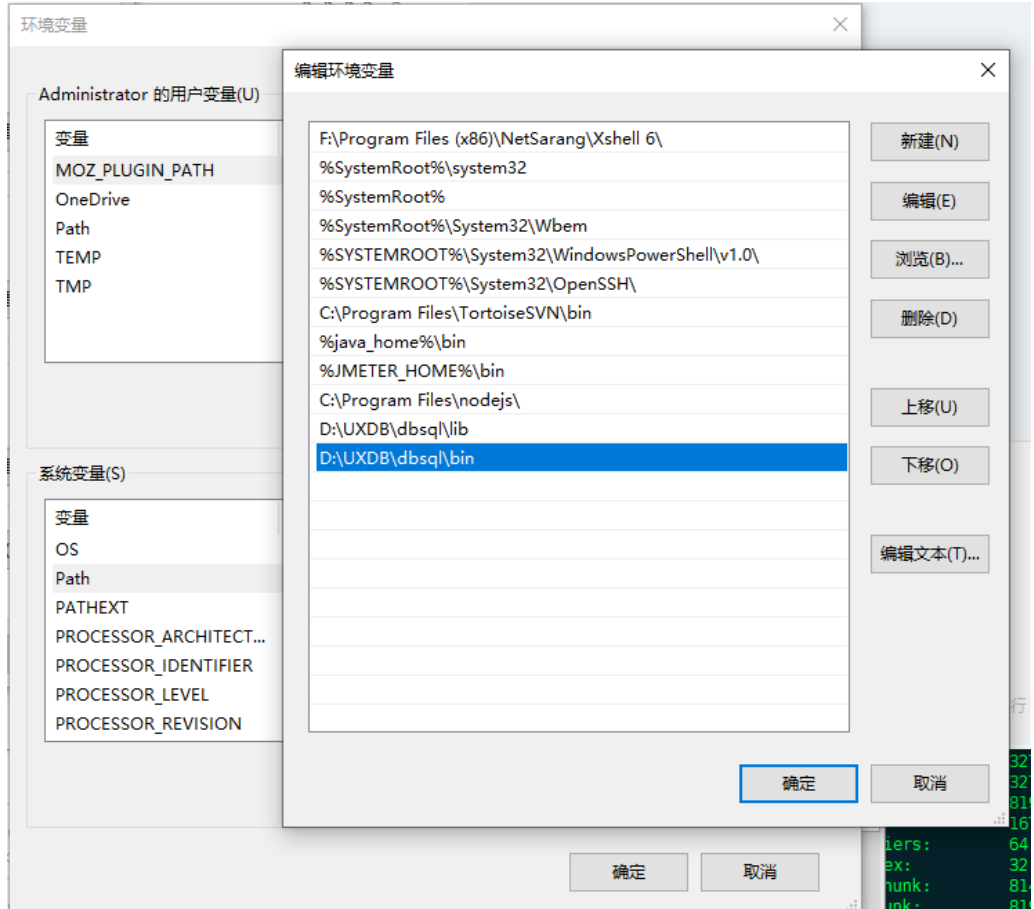




- f. 右键“项目” => “添加” => “现有项”，添加“ssleay32.dll”、“_int.dll”、“libeay32.dll”、“libxsql.dll”到创建的项目文件夹下，根据实际路径添加。



- g. 将uxdb\dbsql\bin目录添加到系统环境变量Path中。



提示

如果执行过程中有关于缺少某个dll文件的报错，需要根据实际情况将文件所在的目录添加到系统环境变量Path中。

h. 右键“项目” => “生成”，成功之后，进行刷新，也可以使用快捷键Ctrl+F5。

4. 运行结果

```

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");

```

datname	datdba	encoding	datcollate	datatype	datistemplate	datallowconn	datconnlimit	datlastsysoid
uxdb	10	6	zh_CN.UTF-8	zh_CN.UTF-8	f	t	-1	13211
mydb	10	6	zh_CN.UTF-8	zh_CN.UTF-8	f	t	-1	13211
template1	10	6	zh_CN.UTF-8	zh_CN.UTF-8	t	t	-1	13211
template0	10	6	zh_CN.UTF-8	zh_CN.UTF-8	t	f	-1	13211
test	10	6	zh_CN.UTF-8	zh_CN.UTF-8	f	t	-1	13211

```

ssh://uxdb:*****@192.168.1.82:22
要添加当前会话，点击左侧的箭头按钮。
1 82 x +
uxdb=# select * from ux_database;
 datname | datdba | encoding | datcollate | datatype | datistemplate | datallowconn | datconnlimit | datlastsysoid |
-----+-----+-----+-----+-----+-----+-----+-----+-----+
 uxdb    | 10     | 6        | zh_CN.UTF-8 | zh_CN.UTF-8 | f             | t             | -1            | 13211         |
 mydb    | 10     | 6        | zh_CN.UTF-8 | zh_CN.UTF-8 | f             | t             | -1            | 13211         |
 template1 | 10    | 6        | zh_CN.UTF-8 | zh_CN.UTF-8 | t             | t             | -1            | 13211         |
 template0 | 10    | 6        | zh_CN.UTF-8 | zh_CN.UTF-8 | t             | f             | -1            | 13211         |
 test    | 10     | 6        | zh_CN.UTF-8 | zh_CN.UTF-8 | f             | t             | -1            | 13211         |
(5 rows)
uxdb=#

```

重要

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=192.168.1.82 port=5432 dbname=uxdb user=uxdb password=123456。

第 3 章 Golang

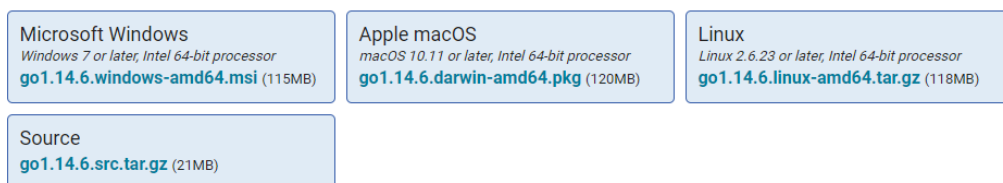
Golang使用UXDB数据库，需要使用uxgo模块来连接，Linux平台也可以使用该模块进行连接。

1. 安装Golang

官网下载安装Golang: <https://golang.google.cn/>

Golang版本要求: Go >=1.10

Featured downloads



官网提供多种平台的安装包，根据实际情况进行下载。

- Windows
 - 下载对应Windows环境的安装包，后缀名为.msi。
 - 双击运行，直接进行下一步，注意GOROOT为GO语言的主目录，GOPATH为工作目录。
 - 默认安装go语言完成后，会自动添加环境变量，若自主修改了GOROOT与GOPATH目录路径，则需要修改环境变量。
 - 完成后可在CMD命令行运行go version来查看版本号信息。
- Linux
 - 下载对应Linux环境的安装包，后缀名为.tar.gz。
 - 执行tar命令解压安装包至/usr/local目录下（官方推荐目录）。
 - 添加/usr/local/go/bin目录到path变量中，在/etc/profile文件最后一行添加：

```
export GOROOT=/usr/local/go
export PATH=$PATH:$GOROOT/bin
```

保存后，执行如下命令：

```
source /etc/profile
```
 - 安装完成之后执行go version，如果出现版本号信息，则Go环境安装成功。

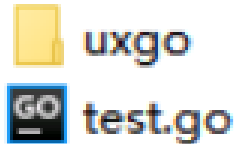
2. 创建环境并连接数据库

- 在GOPATH工作目录下创建工程文件夹。

如：GOPATH目录为E:\go，在E:\go目录下创建一个工程文件夹，如：uxdb_test。

> 本地磁盘 (E:) > go > uxdb_test

名称



- b. 在工程文件夹uxdb_test下创建test.go文件，内容如下。

```
package main
import (
    "database/sql"
    "fmt"
    _ "uxgo" // 操作其实只是引入该包。当导入一个包时，它所有的init()函数就会被执行，但有些时候并非真的需要使用这些包，仅仅是希望它的init()函数被执行而已。这个时候就可以使用_操作引用该包
)
var db *sql.DB
var err error //定义全局变量
func sqlOpen() {
    db, err = sql.Open("UXres", "host=192.168.168.128 port=5432 user=uxdb password=123456 dbname=uxdb sslmode=disable")
    //driverName是数据库驱动类型名称，uxdb的go语言驱动名称为UXres;
    //port是数据库的端口号，默认是5432;
    //user是数据库的登录帐号;
    //dbname是数据库名称;
    //sslmode是安全验证模式，disable为不使用ssl加密方式连接;
    if err != nil {
        fmt.Println("连接数据库错误: " + err.Error())
    } else {
        fmt.Println("数据库连接成功")
    }
    err = db.Ping() //Ping检查与数据库的连接是否有效，如果失效返回error信息
    if err != nil {
        fmt.Println("DBPing错误: " + err.Error())
    } else {
        {
            fmt.Println("DBPingSuccess")
        }
    }
}
func sqlcreate() {
    fmt.Println("*****开始创建表*****")
}
```

```

createsql := "create table Student(Sno int,Sname char(10),Sage int,Ssex char(10));"
stmt, err := db.Prepare(createsql)
if err != nil {
    fmt.Print(err.Error())
} else {
    fmt.Println("create table successfully")
}
stmt.Exec()
}

func sqlInsert() {
    //插入数据
    fmt.Println("+++++开始进行数据库插入+++++")
    stmt, err := db.Prepare("INSERT INTO student(sno,sname,sage,ssex) VALUES($1,$2,$3,$4)
")
    checkErr(err)
    res, err := stmt.Exec("1", "mary", "22", "女")
    checkErr(err)
    affect, err := res.RowsAffected() //RowsAffected返回被update、insert或删除命令影响
的行数
    checkErr(err)
    fmt.Println("受影响的行:", affect)
}

func sqlInsert1() {
    //插入数据
    fmt.Println("+++++开始进行数据库插入+++++")
    stmt, err := db.Prepare("INSERT INTO student(sno,sname,sage,ssex) VALUES($1,$2,$3,$4)
")
    checkErr(err)
    res, err := stmt.Exec("2", "jack", "23", "男")
    checkErr(err)
    affect, err := res.RowsAffected() //RowsAffected返回被update、insert或删除命令影响
的行数
    checkErr(err)
    fmt.Println("受影响的行:", affect)
}

func sqlDelete() {
    //删除数据
    fmt.Println("=====开始删除数据=====")
    stmt, err := db.Prepare("delete from student where sno=$1")
    checkErr(err)
    res, err := stmt.Exec(1)
    checkErr(err)
    affect, err := res.RowsAffected()
    checkErr(err)
    fmt.Println("受影响的行:", affect)
}

func sqlSelect() {
    //查询数据
    fmt.Println("*****开始进行数据库查询*****")
}

```

```
rows, err := db.Query("SELECT * FROM student")
checkErr(err)
for rows.Next() {
    var sno int
    var sname string
    var sage int
    var ssex string
    err = rows.Scan(&sno, &sname, &sage, &ssex)
    checkErr(err)
    fmt.Println("sno = ", sno, "\nsname = ", sname, "\nsage = ", sage, "\nssex = ", ssex,
"\n-----")
}
}

func sqlUpdate() {
    //更新数据
    fmt.Println("#####开始进行数据更新#####")
    stmt, err := db.Prepare("update student set sname=$1 where sno=$2")
    checkErr(err)
    res, err := stmt.Exec("uxsino", 1)
    checkErr(err)
    affect, err := res.RowsAffected()
    checkErr(err)
    fmt.Println("受影响的行:", affect)
}

func sqlClose() {
    //关闭数据库
    db.Close()
}

func checkErr(err error) {
    //定义异常排查函数
    if err != nil {
        panic(err)
    }
}

func sqlTest() {
    //创建测试函数，调用增删改查函数进行数据库操作
    sep := "-----\n"

    sqlInsert()
    sqlInsert1()
    sqlSelect()

    sqlUpdate()
    sqlSelect()

    sqlDelete()
    sqlSelect()

    sqlClose()
    println(sep, "数据库已关闭")
}
```

```
func main() {  
    //运行打开数据库函数  
    sqlOpen()  
    //运行创建表函数  
    sqlcreate()  
    //运行增删改查测试函数  
    sqlTest()  
}
```

- c. 将GO语言数据库驱动uxgo分别放在test.go的同级目录和GOROOT的src目录下。

GOROOT目录: windows一般在C:\Go\src目录下; Linux一般在/usr/local/go/src目录下。根据实际安装golang时的环境变量而定。

- d. 运行结果

- Windows: 打开cmd窗口, 进入test.go所在的目录, 执行命令go run test.go运行。
- Linux: 打开终端, 进入test.go所在的目录, 执行命令go run test.go运行。

```
E:\go\uxdb_test>go run test.go
数据库连接成功
DBPingSuccess
*****开始创建表*****
create table successfully
+++++开始进行数据库插入+++++
受影响的行: 1
+++++开始进行数据库插入+++++
受影响的行: 1
*****开始进行数据库查询*****
sno = 1
sname = mary
sage = 22
ssex = 女
-----
sno = 2
sname = jack
sage = 23
ssex = 男
-----
#####开始进行数据更新#####
受影响的行: 1
*****开始进行数据库查询*****
sno = 2
sname = jack
sage = 23
ssex = 男
-----
sno = 1
sname = uxsino
sage = 22
ssex = 女
-----
=====开始删除数据=====
受影响的行: 1
*****开始进行数据库查询*****
sno = 2
sname = jack
sage = 23
ssex = 男
-----
-----
数据库已关闭
```

第 4 章 JDBC

1. 安装jdk

安装jdk7及以上版本。打开cmd窗口，输入`java -version`查看已安装的jdk。

2. 配置eclipse

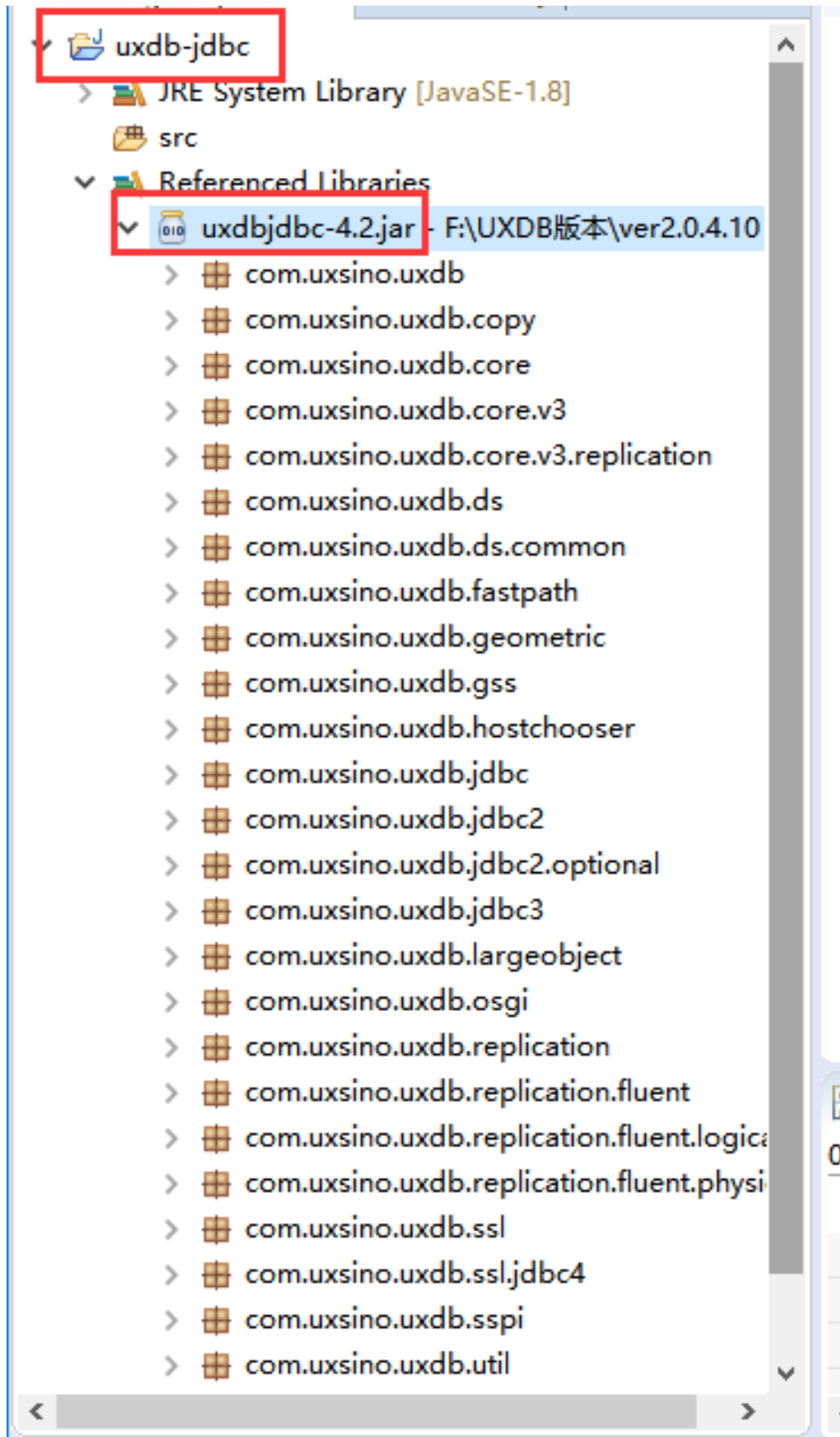
a. 新建java project

打开eclipse，新建名为uxdb-jdbc的java project（File=>New=>Java Project）。

b. 导入uxdb的jdbc包

将uxdb的jdbc包（uxdbjdbc-4.1.jar/uxdbjdbc-4.2.jar在uxdb安装目录中获取，其中JDK1.7使用uxdbjdbc-4.1.jar，JDK1.8使用uxdbjdbc-4.2.jar）添加到项目的外部引用库中。

（右键uxdb-jdbc，Build Path=>Add External Archives...）



c. 创建表student

数据库用户名uxdb、密码123456，创建新的数据库test，在test中创建表student并插入数据，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;
```

d. 新建class

新建两个名为ConnUtil和ConnTest的class，指定包名为jdbc。

uxdb所在机器IP为192.168.1.82，集群port为5432，数据库名为test，用户名为uxdb，密码为123456。



ConnUtil.java代码如下：

```
package jdbc;
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.uxsino.uxdb.Driver");
        String url = "jdbc:uxdb://192.168.1.82:5432/test";
        try
        {
            conn = DriverManager.getConnection(url, "uxdb", "123456");
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
    catch (ClassNotFoundException e)
    {
        e.printStackTrace();
    }
    return conn;
}
}
```

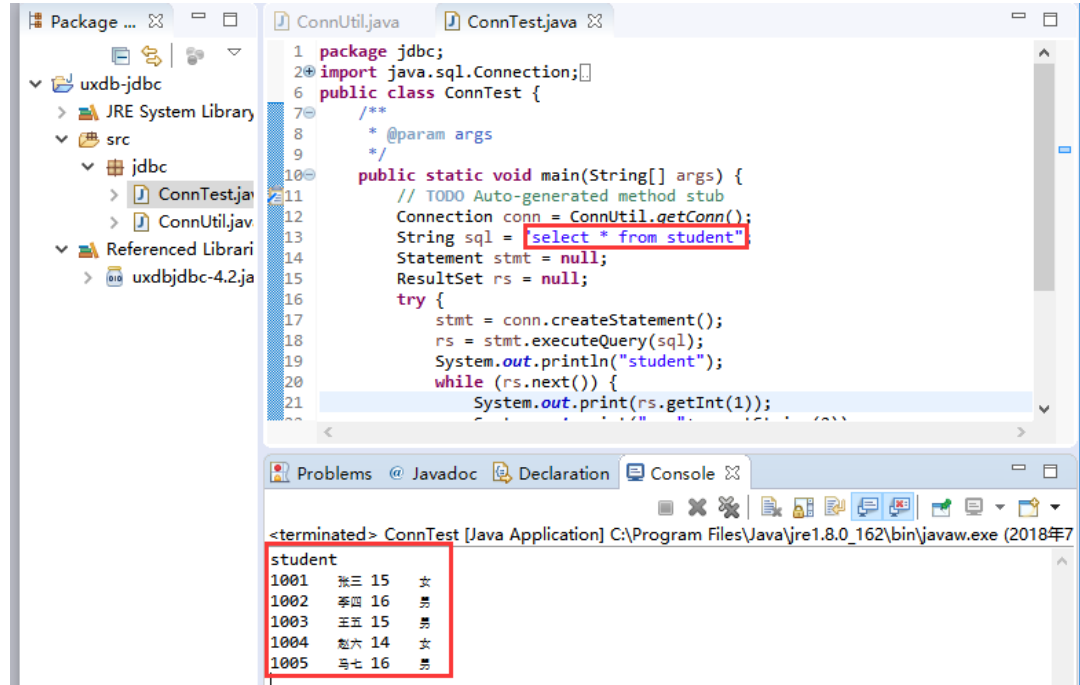
ConnTest.java代码如下:

```
package jdbc;
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();
        }
    }
}
```

```
}  
}  
}
```

3. 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。



The screenshot shows an IDE with two files: ConnUtil.java and ConnTest.java. ConnTest.java contains the following code:

```
1 package jdbc;  
2 import java.sql.Connection;[]  
6 public class ConnTest {  
7     /**  
8      * @param args  
9      */  
10    public static void main(String[] args) {  
11        // TODO Auto-generated method stub  
12        Connection conn = ConnUtil.getConnection();  
13        String sql = "select * from student";  
14        Statement stmt = null;  
15        ResultSet rs = null;  
16        try {  
17            stmt = conn.createStatement();  
18            rs = stmt.executeQuery(sql);  
19            System.out.println("student");  
20            while (rs.next()) {  
21                System.out.print(rs.getInt(1));  
22            }  
23        } catch (SQLException e) {  
24            e.printStackTrace();  
25        }  
26    }  
27 }
```

The console output shows the following data:

```
<terminated> ConnTest [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (2018年7  
student  
1001 张三 15 女  
1002 李四 16 男  
1003 王五 15 男  
1004 赵六 14 女  
1005 马七 16 男
```

```

ssh://192.168.1.82:22
要添加当前会话，点击左侧的箭头按钮。
1 192.168.1.82:22 x +
Password:
The license due date is: 2028-12-01 00:00:00
It's commercial license.
uxsql (10.0)
Type "help" for help.

test=# select * from student ;
 sno | sname | sage | ssex
-----+-----+-----+-----
 1001 | 张三  |   15 | 女
 1002 | 李四  |   16 | 男
 1003 | 王五  |   15 | 男
 1004 | 赵六  |   14 | 女
 1005 | 马七  |   16 | 男
(5 rows)

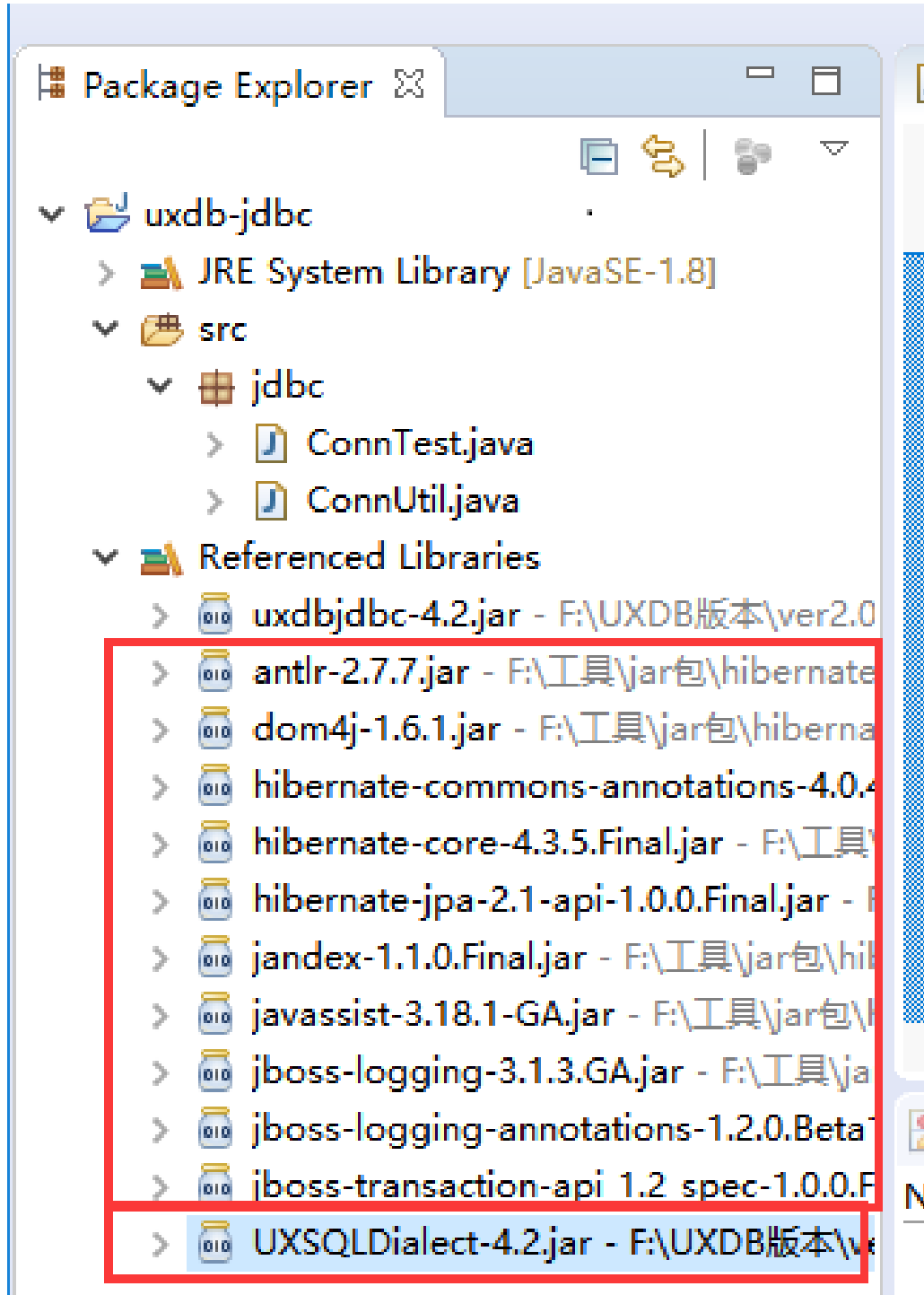
```

4. 使用分析说明
 - a. `java.sql.Connection`连接数据库类。
 - b. `java.sql.DriverManager`驱动管理类。
 - c. `Class.forName("com.uxsino.uxdb.Driver")`调用uxdb的jdbc包中的Driver.class。
 - d. `DriverManager.getConnection("jdbc:uxdb://192.168.1.82:5432/test", "uxdb", "123456")`参数格式"jdbc:uxdb://IP:port/databasename", "username", "password"。

4.1. Hibernate

1. 安装hibernate tools

官网下载离线包或者在线安装hibernate插件（JBoss Tools）。
2. 导入hibernate驱动包
 - a. 导入官网下载的hibernate驱动包；
 - b. 导入uxdb的hibernate驱动包uxdb的hibernate驱动包（UXSQLDialect-4.2.jar）在uxdb的安装目录中获取。

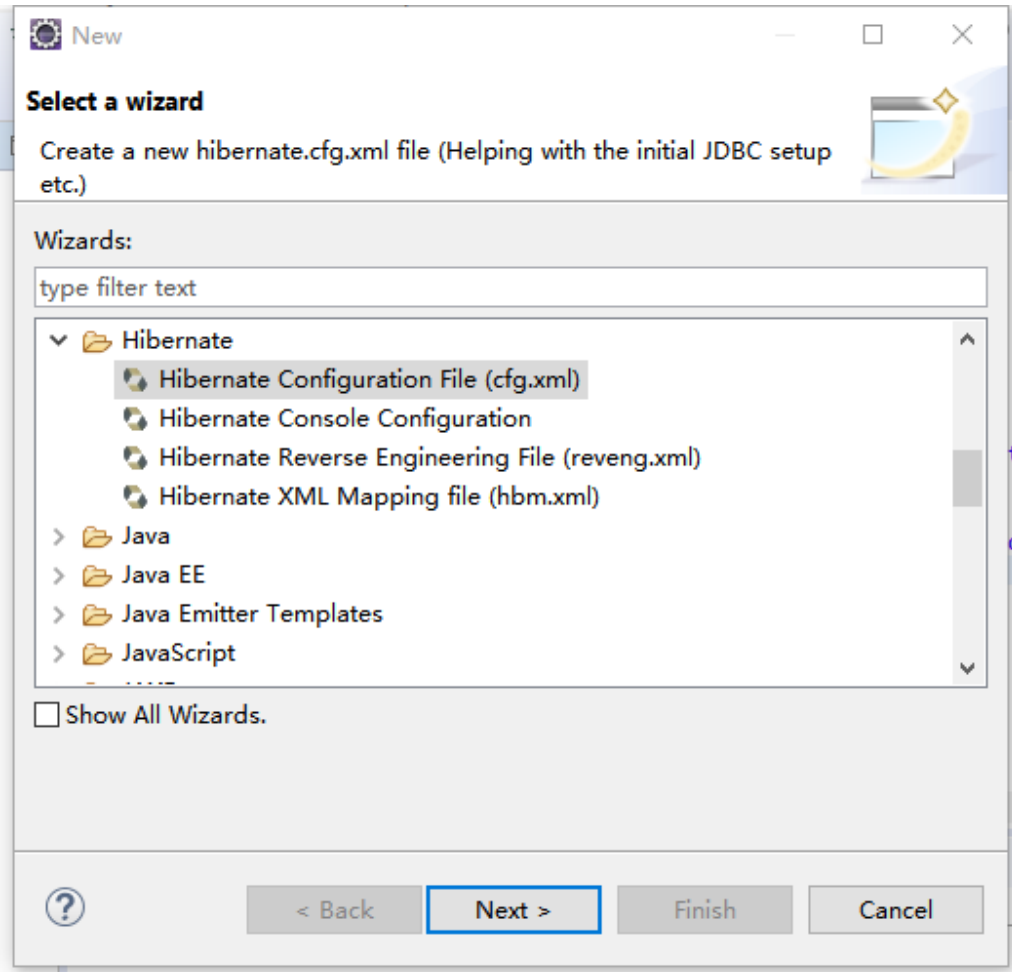


3. 创建hibernate.cfg.xml

uxdb server端IP为192.168.1.82, 端口为5432, 数据库为test, 用户名为uxdb, 密码为123456。

a. 创建hibernate.cfg.xml;

右键src, New=>Other...=>Hibernate=>Hibernate Configuration File(cfg.xml), 单击Next。



根据下图填写相关配置，勾选Create a console configuration，单击Next：

Session factory name: uxdb

Database dialect: org.hibernate.dialect.UXSQLDialect

Driver class: com.uxsino.uxdb.Driver

Connection URL: jdbc:uxdb://192.168.1.82:5432/test

Username: uxdb

Password: 123456

Hibernate Configuration File (cfg.xml)

This wizard creates a new configuration file to use with Hibernate.

Container: /uxdb-jdbc/src

File name: hibernate.cfg.xml

Hibernate version: 5.2

Session factory name: uxdb

[Get values from Connection](#)

Database dialect: org.hibernate.dialect.UXSQLDialect

Driver class: com.uxsino.uxdb.Driver

Connection URL: jdbc:uxdb://192.168.1.82:5432/test

Default Schema:

Default Catalog:

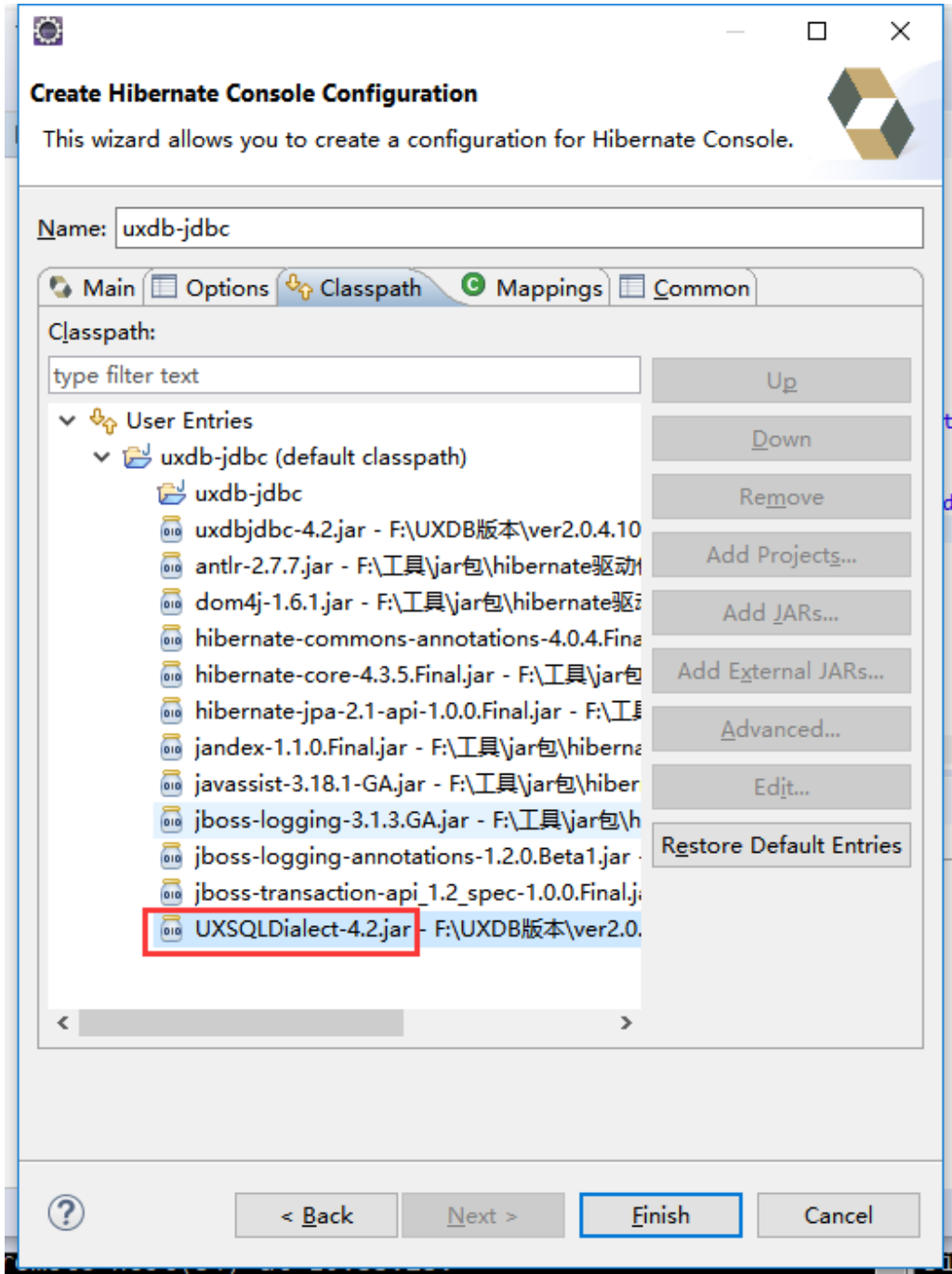
Username: uxdb

Password: 123456

Create a console configuration

[?](#) < Back Next > Finish Cancel

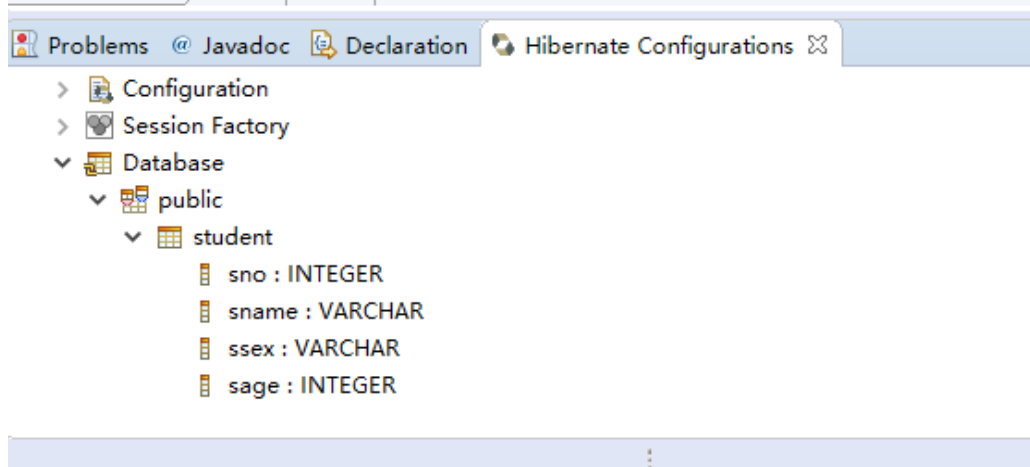
- b. 单击Classpath标签页，选择uxdb的JDBC驱动包，然后Finish。



4. 连接数据库并生成源代码

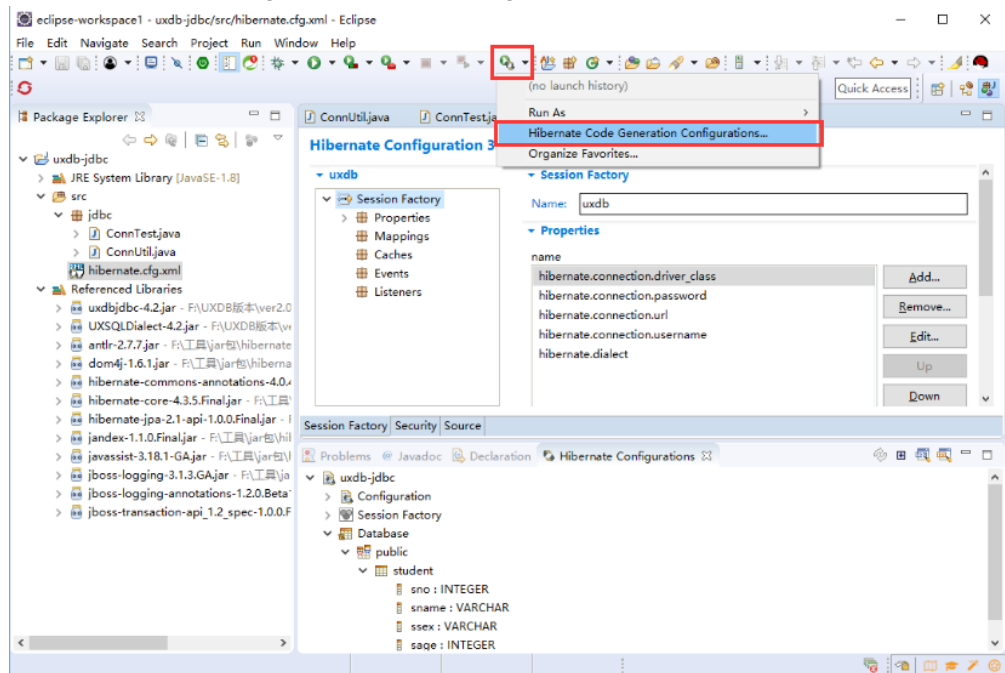
- 打开Hibernate Configurations的View窗口，如下图显示即数据库连接成功。

Window=>Show View=>other..=>Hibernate=>Hibernate Configurations, 单击open。

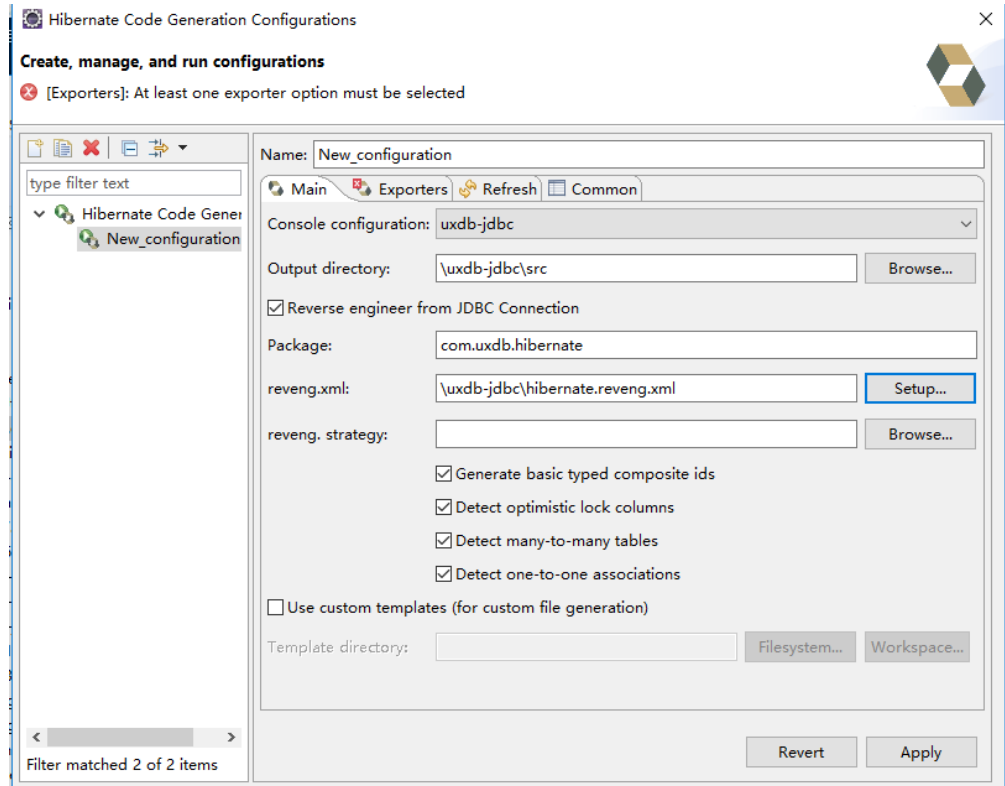


b. 反向生成java代码

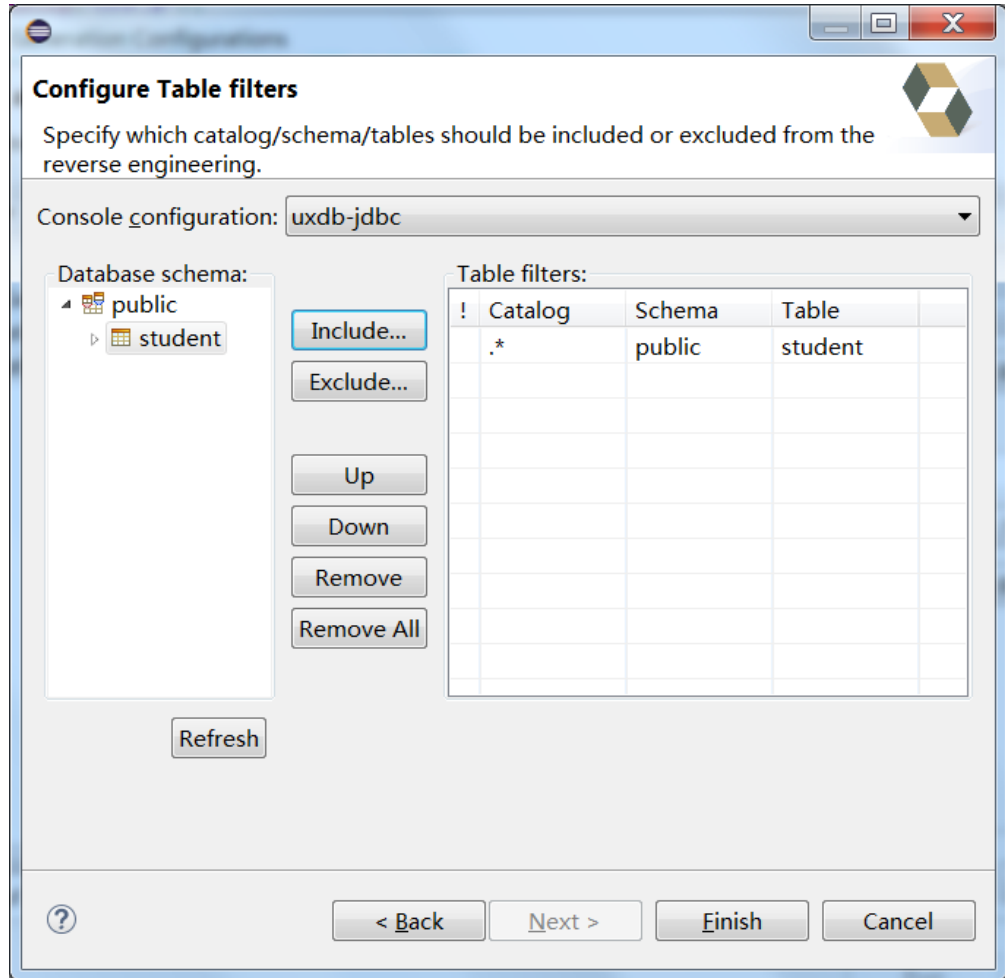
- 新建Hibernate Code generation 的configure



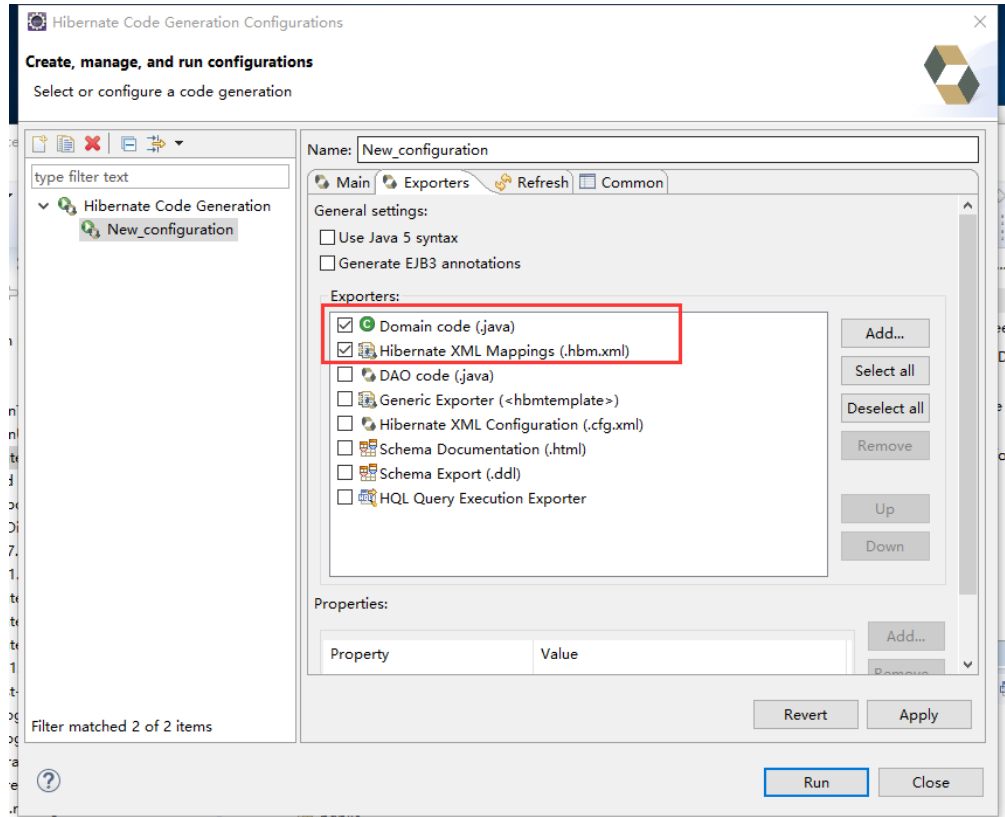
双击Hibernate Code Generation打开配置文件，名称默认是New_configuration；Output directory选择\uxdb-jdbc\src；勾选Reverse engineer from JDBC Connection；Package填写com.uxdb.hibernate



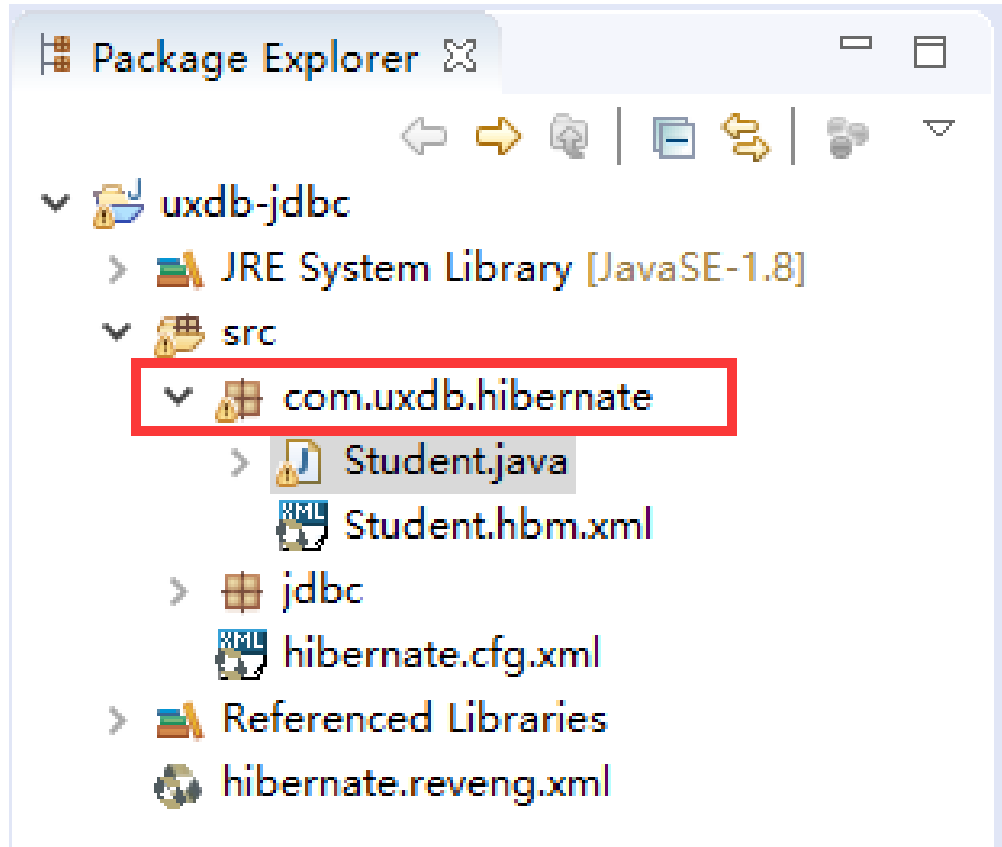
- reveng.xml单击setup...=>Create new..., 选择路径uxdb-jdbc, 单击Next, 进入Configure Table filters页面, 单击Refresh, 点开public, 选择student, 单击Include...
- 单击Finish。



- 选择Exporters选项卡，并进行如下的配置，勾选Domain code和Hibernate XML Mappings，单击Run完成。



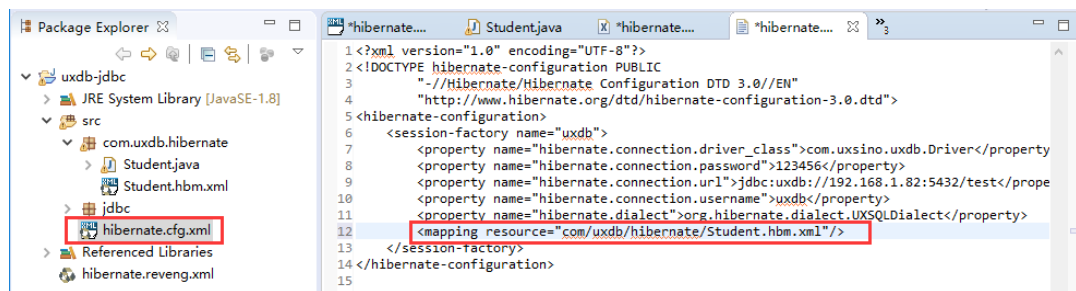
- 工程中生成了com.uxdb.hibernate包及访问数据库的源代码，如下所示。



5. 修改hibernate.cfg.xml

右键hibernate.cfg.xml, Open With=>Test Editor, 增加如下内容:

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



6. 创建hibernate的测试类

右键包com.uxdb.hibernate, New=>class, Name输入HibernateTest, 单击Finish。

HibernateTest.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", "deprecation" })
    public static void main(String[] args) {
        //读取hibernate.cfg.xml文件
        Configuration cfg = new Configuration().configure();
        //建立SessionFactory
        ServiceRegistry sr = new
        ServiceRegistryBuilder().applySettings(cfg.getProperties()).buildServiceRegistry();
        SessionFactory factory = cfg.buildSessionFactory(sr);
        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();
                }
            }
        }
    }
}
```

7. 运行结果

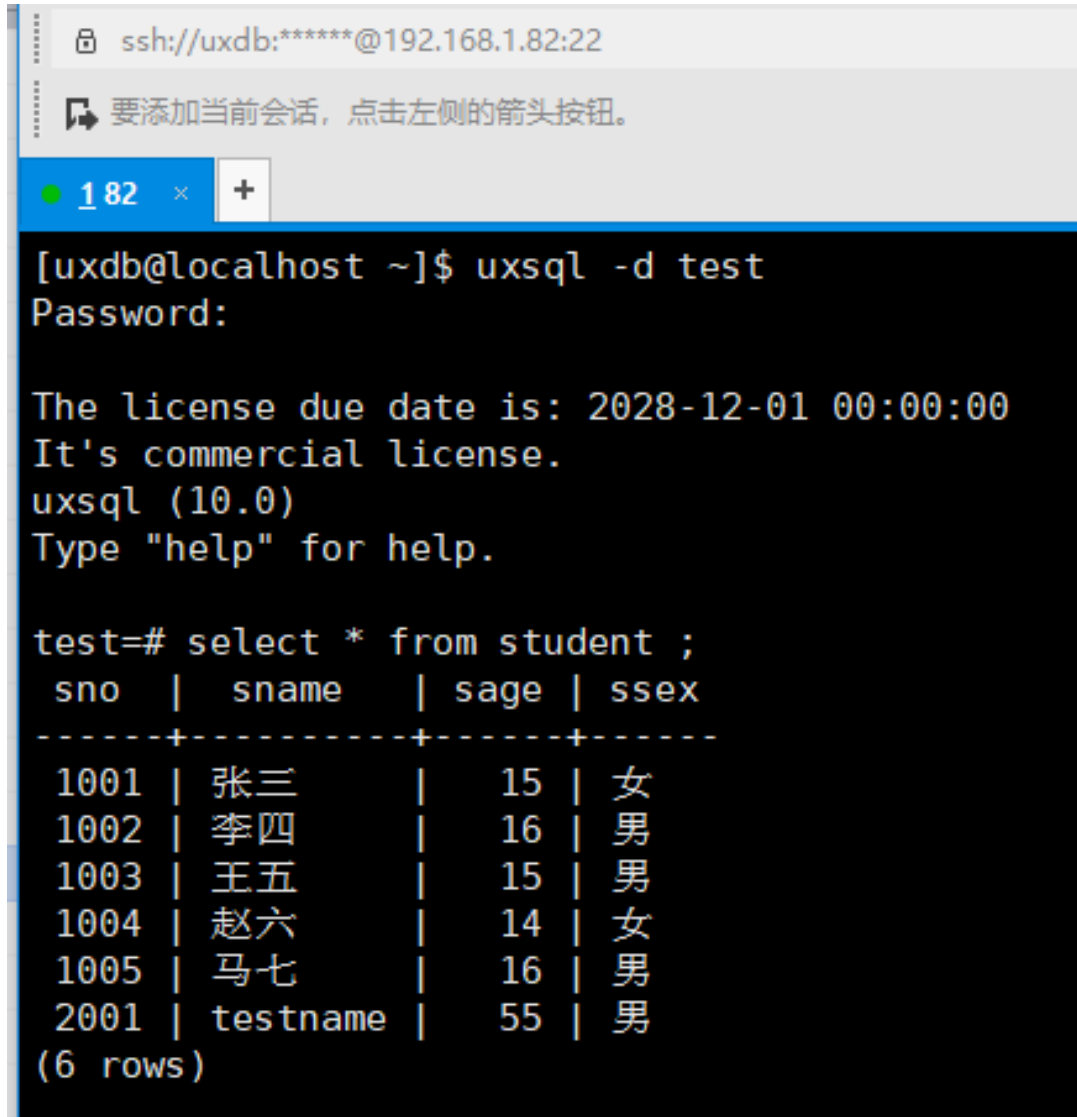
运行结果如下显示时hibernate连接使用uxdb成功，显示student表的数据与uxdb数据库中查看结果一致。


```
16 //取得session
17 session = factory.openSession();
18 //开始事务
19 session.beginTransaction();
20 Student student = new Student();
21 student.setSno(2001);
22 student.setName("testname");
23 student.setSage(55);
24 student.setSsex("男");
25 //保存student对象
26 session.save(student);
27 List<Student> list;
28 String sql = "from Student";
29 Query query = session.createQuery(sql);
30 list = (List<Student>)query.list();
31 System.out.println("studentnew");
32 for (Student u : list) {
33     System.out.println(u.getSno() + ";" + u.getName() + ";" + u.getSage() + ";" + u.getSex());
34 }
35 //提交事务
```

Problems @ Javadoc Declaration Console Hibernate Configurations

HibernateTest [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (2018年7月11日 下午2:35:14)

```
studentnew
1001;张三;15;女
1002;李四;16;男
1003;王五;15;男
1004;赵六;14;女
1005;马七;16;男
2001;testname;55;男
```



```
ssh://uxdb:*****@192.168.1.82:22
要添加当前会话，点击左侧的箭头按钮。
182 × +
[uxdb@localhost ~]$ uxsql -d test
Password:

The license due date is: 2028-12-01 00:00:00
It's commercial license.
uxsql (10.0)
Type "help" for help.

test=# select * from student ;
 sno |  sname  |  sage |  ssex
-----+-----+-----+-----
 1001 |  张三   |    15 |  女
 1002 |  李四   |    16 |  男
 1003 |  王五   |    15 |  男
 1004 |  赵六   |    14 |  女
 1005 |  马七   |    16 |  男
 2001 |  testname |    55 |  男
(6 rows)
```

8. 使用分析说明

创建配置hibernate.cfg.xml时注意以下几项内容：

- Database dialect: org.hibernate.dialect.UXSQLDialect (使用uxdb的dialect)。
- Driver class: com.uxsino.uxdb.Driver (使用uxdb的driver.class)。
- Connection url: jdbc:uxdb://192.168.1.82:5432/test (jdbc:uxdb://IP:port/databasename)。
- Username: uxdb。
- Password: 123456。

注意

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。

4.2. Spring

1. 安装SpringIDE

下载离线包或者在线安装SpringIDE插件。

2. 导入spring所有驱动包

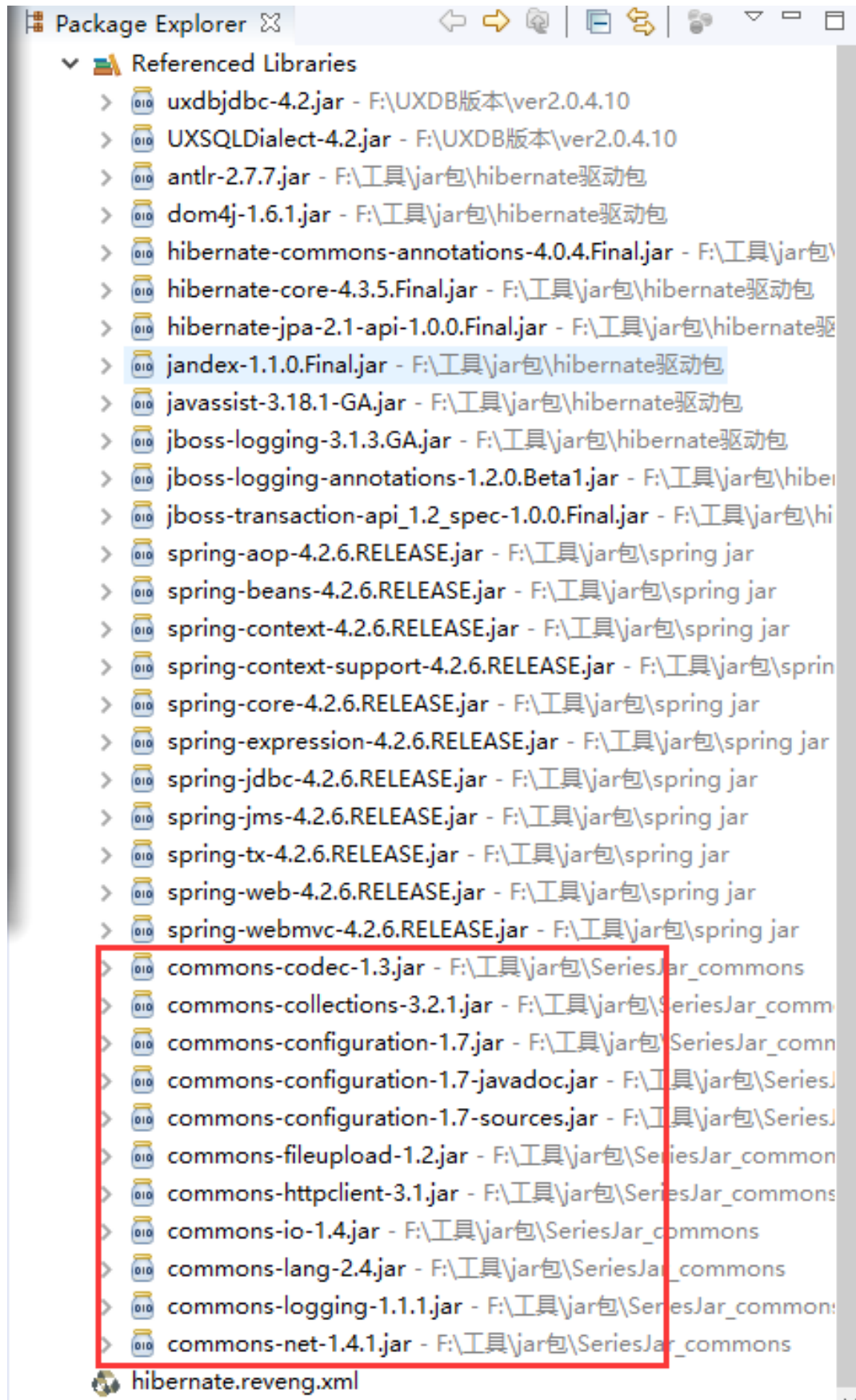
导入官网下载的spring驱动包。

The screenshot shows the Package Explorer for a project named 'uxdb-jdbc'. The project structure is as follows:

- uxdb-jdbc
 - JRE System Library [JavaSE-1.8]
 - src
 - com.uxdb.hibernate
 - jdbc
 - hibernate.cfg.xml
 - spring
 - com.uxdb.bean
 - SpringJdbcTest.java
 - com.uxdb.service
 - StudentBean.java
 - com.uxdb.service.impl
 - StudentService.java
 - com.uxdb.springtest
 - StudentServiceImpl.java
 - Referenced Libraries
 - uxdbjdbc-4.2.jar - F:\UXDB版本\ver2.0.4.10
 - UXSQLDialect-4.2.jar - F:\UXDB版本\ver2.0.4.10
 - antlr-2.7.7.jar - F:\工具\jar包\hibernate驱动包
 - dom4j-1.6.1.jar - F:\工具\jar包\hibernate驱动包
 - hibernate-commons-annotations-4.0.4.Final.jar - F:\工具\jar包\hibernate驱动包
 - hibernate-core-4.3.5.Final.jar - F:\工具\jar包\hibernate驱动包
 - hibernate-jpa-2.1-api-1.0.0.Final.jar - F:\工具\jar包\hibernate驱动包
 - jandex-1.1.0.Final.jar - F:\工具\jar包\hibernate驱动包
 - javassist-3.18.1-GA.jar - F:\工具\jar包\hibernate驱动包
 - jboss-logging-3.1.3.GA.jar - F:\工具\jar包\hibernate驱动包
 - jboss-logging-annotations-1.2.0.Beta1.jar - F:\工具\jar包\hibernate驱动包
 - jboss-transaction-api_1.2_spec-1.0.0.Final.jar - F:\工具\jar包\hibernate驱动包
 - spring-aop-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-beans-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-context-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-context-support-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-core-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-expression-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-jdbc-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-jms-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-tx-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-web-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - spring-webmvc-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - hibernate.reveng.xml

3. 导入commons驱动包

导入官网下载的commons驱动包。



4. 创建Bean类、接口类、接口实现类、beans.xml配置和测试类

右键uxdb-jdbc, New=>Source Folder, Folder name输入spring。

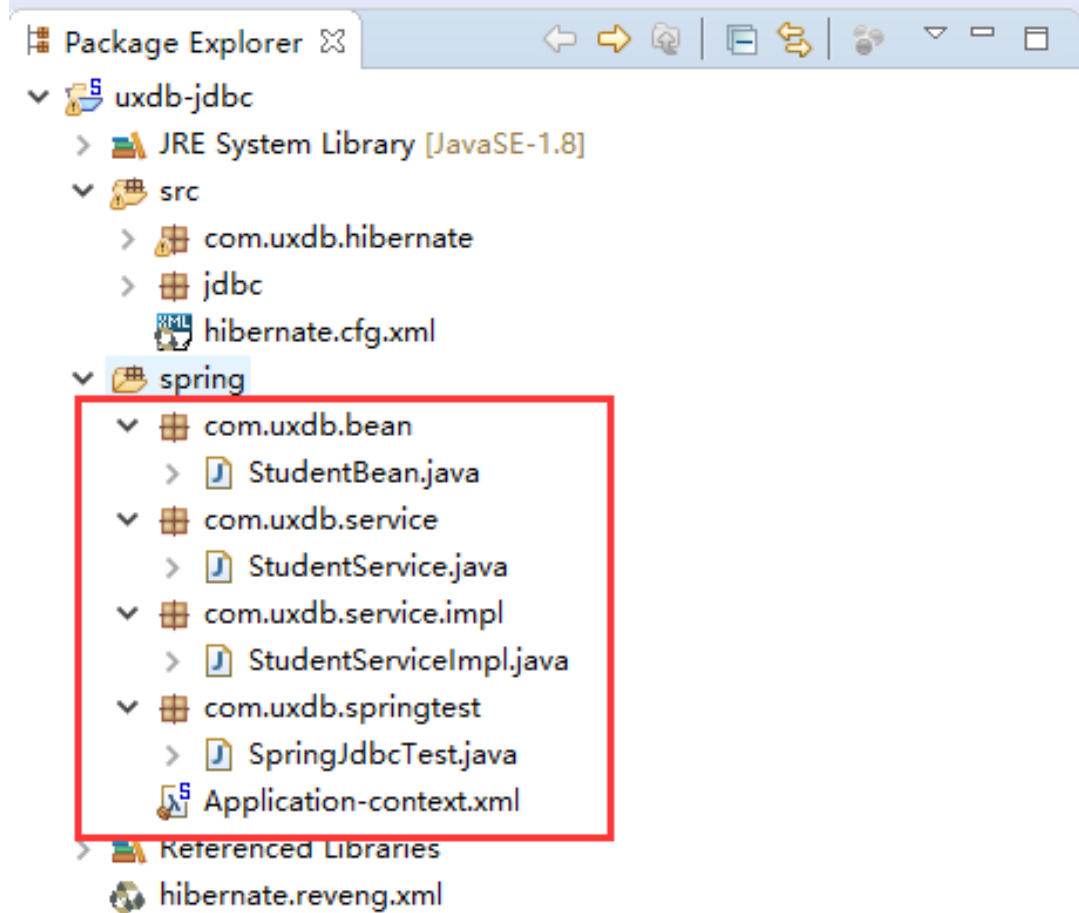
创建包（右键spring, New=>Package）

com.uxdb.bean, com.uxdb.service, com.uxdb.service.impl, com.uxdb.springtest。

包下面创建对应的类（右键包名, New=>Class）

StudentBean, StudentService, StudentServiceImpl, SpringJdbcTest。

创建Spring Bean配置文件（右键spring, New=>Other...=>Spring=>Spring Bean Configuration File, 单击Next, File name中输入Application-context, 单击Finish）。



a. 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;  
}  
}
```

b. 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);  
}
```

c. 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 });
    }

```

```
}
}
```

- d. 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.uxsino.uxdb.Driver"/>
    <property name="url" value="jdbc:uxdb://192.168.1.82:5432/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>
```

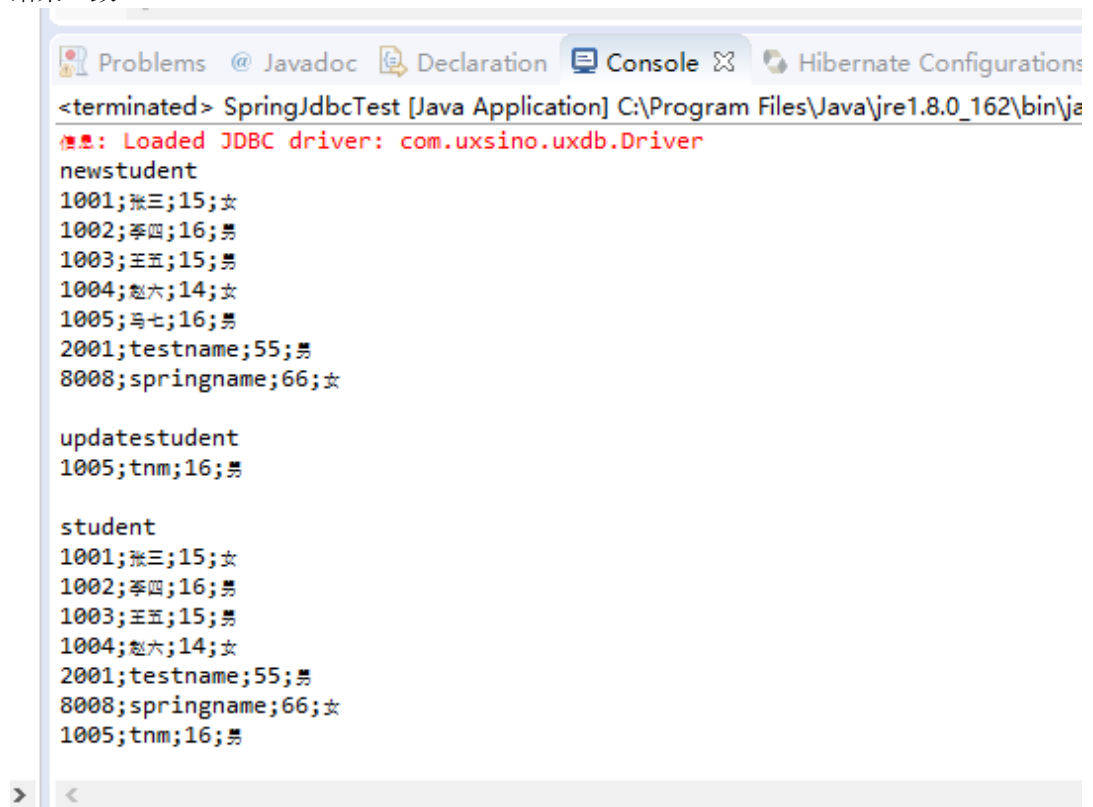
- e. SpringJdbcTest.java;

```
package com.uxdb.springtest;
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.setName("tnm");
student.setSsex("男");
studentService.update(student);
student = studentService.getStudent(1005);
System.out.println("");
System.out.println("updatestudent");
System.out.println(student.getSno() + ";" + student.getName() + ";" + 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.getName() + ";" + u.getSage() + ";" + u.getSsex());
}
}
```

5. 运行结果

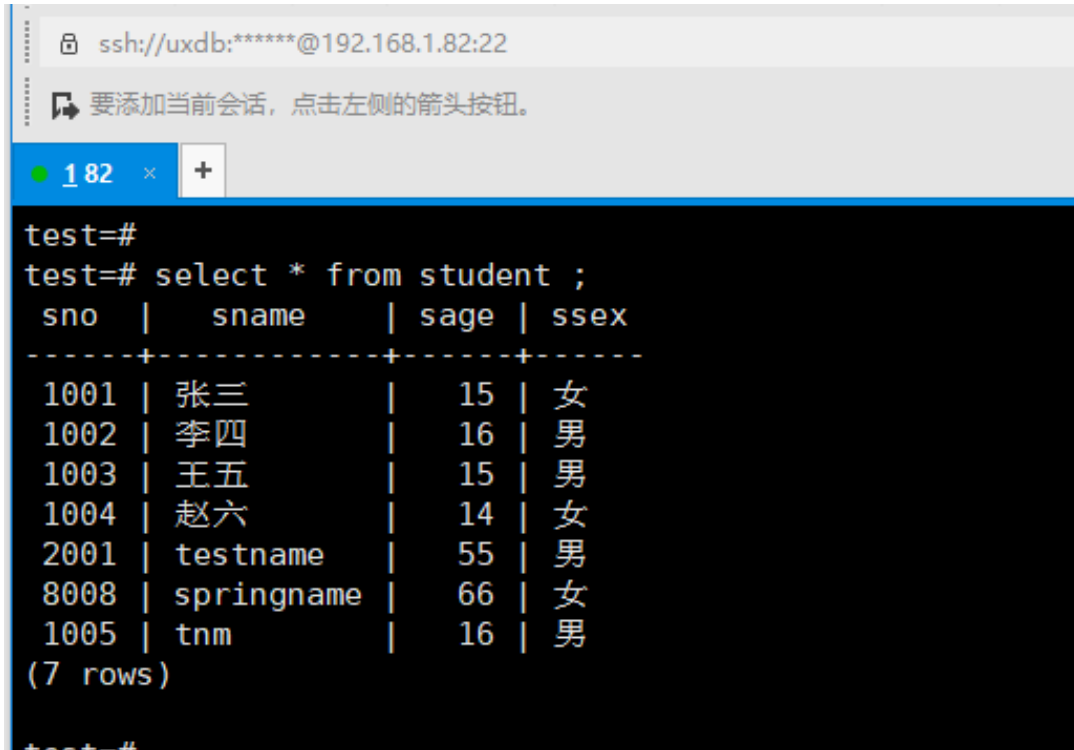
运行结果如下显示时spring连接使用uxdb成功，显示student表的数据与uxdb数据库中查看结果一致。



```
<terminated> SpringJdbcTest [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\ja
Loaded JDBC driver: com.uxsino.uxdb.Driver
newstudent
1001;张三;15;女
1002;李四;16;男
1003;王五;15;男
1004;赵六;14;女
1005;马七;16;男
2001;testname;55;男
8008;springname;66;女

updatestudent
1005;tnm;16;男

student
1001;张三;15;女
1002;李四;16;男
1003;王五;15;男
1004;赵六;14;女
2001;testname;55;男
8008;springname;66;女
1005;tnm;16;男
```



The screenshot shows an SSH terminal window with the following content:

```
ssh://uxdb:*****@192.168.1.82:22
要添加当前会话，点击左侧的箭头按钮。
182 x +
test=#
test=# select * from student ;
 sno |  sname  |  sage | ssex
-----+-----+-----+-----
 1001 |  张三   |    15 |  女
 1002 |  李四   |    16 |  男
 1003 |  王五   |    15 |  男
 1004 |  赵六   |    14 |  女
 2001 | testname |    55 |  男
 8008 | springname |    66 |  女
 1005 | tnm     |    16 |  男
(7 rows)
test=#
```

6. 使用分析说明

配置Application-context.xml文件注意以下几点:

- driverClassName="com.uxsino.uxdb.Driver" (使用uxdb的driver.class)。
- url="jdbc:uxdb://192.168.1.223:5432/test" (jdbc:uxdb://IP:port/databasename)。
- username="uxdb"。
- password="123456"。











































4.3. MyBatis

1. 安装MyBatis

下载离线包或者在线安装插件。

2. 导入mybatis的驱动包

导入官网下载的mybatis驱动包。

- ▼  uxdb-jdbc
 - >  JRE System Library [JavaSE-1.8]
 - >  src
 - >  spring
 -  mybatis
 - ▼  Referenced Libraries
 -  uxdbjdbc-4.2.jar - F:\UXDB版本\ver2.0.4.10
 - >  UXSQLDialect-4.2.jar - F:\UXDB版本\ver2.0.4.10
 - >  antlr-2.7.7.jar - F:\工具\jar包\hibernate驱动包
 - >  dom4j-1.6.1.jar - F:\工具\jar包\hibernate驱动包
 - >  hibernate-commons-annotations-4.0.4.Final.jar - F:\工具\jar包\hibernate驱动包
 - >  hibernate-core-4.3.5.Final.jar - F:\工具\jar包\hibernate驱动包
 - >  hibernate-jpa-2.1-api-1.0.0.Final.jar - F:\工具\jar包\hibernate驱动包
 - >  jandex-1.1.0.Final.jar - F:\工具\jar包\hibernate驱动包
 - >  javassist-3.18.1-GA.jar - F:\工具\jar包\hibernate驱动包
 - >  jboss-logging-3.1.3.GA.jar - F:\工具\jar包\hibernate驱动包
 - >  jboss-logging-annotations-1.2.0.Beta1.jar - F:\工具\jar包\hibernate驱动包
 - >  jboss-transaction-api_1.2_spec-1.0.0.Final.jar - F:\工具\jar包\hibernate驱动包
 - >  spring-aop-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-beans-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-context-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-context-support-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-core-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-expression-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-jdbc-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-jms-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-tx-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-web-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  spring-webmvc-4.2.6.RELEASE.jar - F:\工具\jar包\spring jar
 - >  commons-codec-1.3.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-collections-3.2.1.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-configuration-1.7.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-configuration-1.7-javadoc.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-configuration-1.7-sources.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-fileupload-1.2.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-httpclient-3.1.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-io-1.4.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-lang-2.4.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-logging-1.1.1.jar - F:\工具\jar包\SeriesJar_commons
 - >  commons-net-1.4.1.jar - F:\工具\jar包\SeriesJar_commons
 -  mybatis-3.4.4.jar - F:\工具\jar包\MyBatis 3.4.4
 -  hibernate.reveng.xml

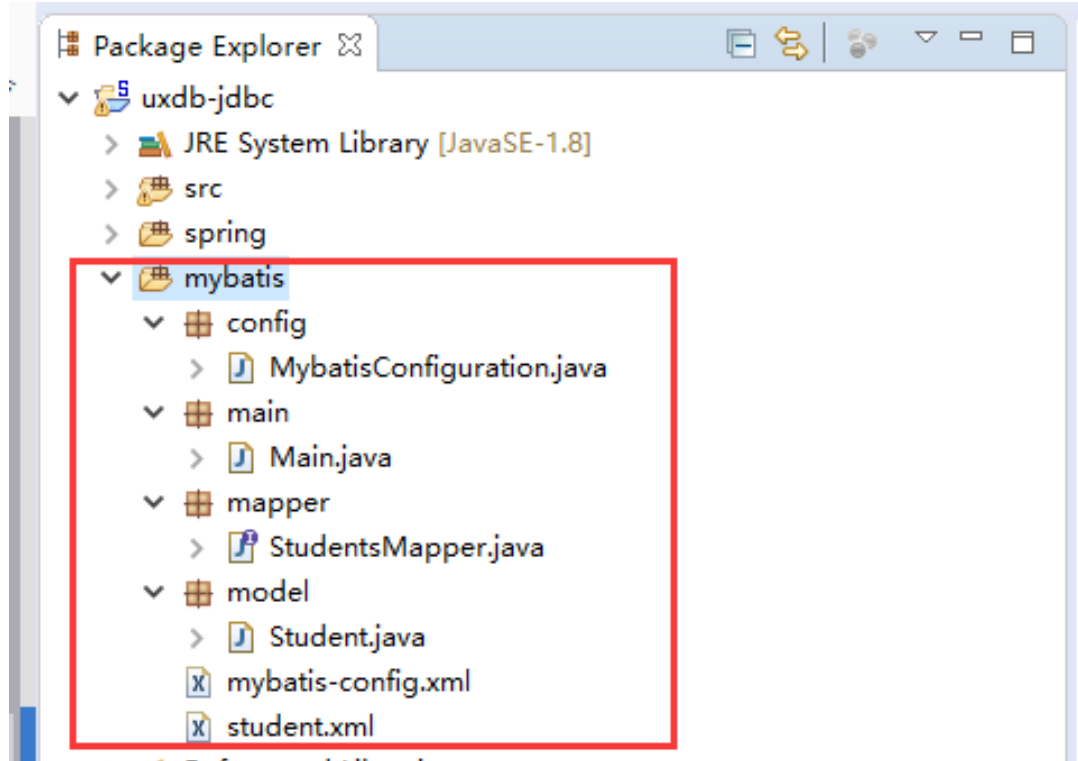
3. 创建接口类、配置和测试类等

右键uxdb-jdbc, New=>Source Folder, Folder name输入mybatis。

创建包（右键mybatis, New=>Package）config, main, mapper, model。

包下面创建对应的类（右键包名, New=>Class）MybatisConfiguration, Main, StudentsMapper, Student。

创建Mybatis配置文件（右键mybatis, New=>Other...=>MyBatis=>MyBatis Generator Configuration File, 单击Next, File name中输入mybatis-config.xml和student.xml, 单击Finish）。



a. Student.java代码如下：

```
package model;
public class Student {
    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 setName(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;
    }
}

```

- b. StudentsMapper代码如下:

```

package mapper;
import model.Student;
import java.util.List;
// Maps queries in student.xml
public interface StudentsMapper {
    public List<Student> getStudents();
}

```

- c. MybatisConfiguration.java代码如下:

```

package config;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import java.io.IOException;
import java.io.Reader;
public class MybatisConfiguration {
    private static final String MYBATIS_CONFIG_FILE = "mybatis-config.xml";
    private final SqlSessionFactory sqlSessionFactory;
    public MybatisConfiguration() throws IOException {
        try(Reader reader = Resources.getResourceAsReader(MYBATIS_CONFIG_FILE)) {
            sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);
        }
    }
    public SqlSessionFactory getSessionFactory() {
        return sqlSessionFactory;
    }
}

```

- d. mybatis-config.xml如下:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://
mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="poolMaximumActiveConnections" value="50"/>
        <property name="poolMaximumIdleConnections" value="10"/>
        <property name="poolMaximumCheckoutTime" value="20000"/> <!--
milliseconds -->
        <property name="driver" value="com.uxsino.uxdb.Driver"/>
        <property name="url" value="jdbc:uxdb://192.168.1.82:5432/test"/>
        <property name="username" value="uxdb"/>
        <property name="password" value="123456"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="student.xml"/>
  </mappers>
</configuration>

```

e. student.xml 如下：

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/
mybatis-3-mapper.dtd">
<mapper namespace="mapper.StudentsMapper">
  <select id="getStudents" parameterType="int" resultType="model.Student">
    INSERT INTO student VALUES(9999,'mybatis',99,'boy');
    SELECT * FROM student
  </select>
</mapper>

```

f. 测试代码Main.java 如下：

```

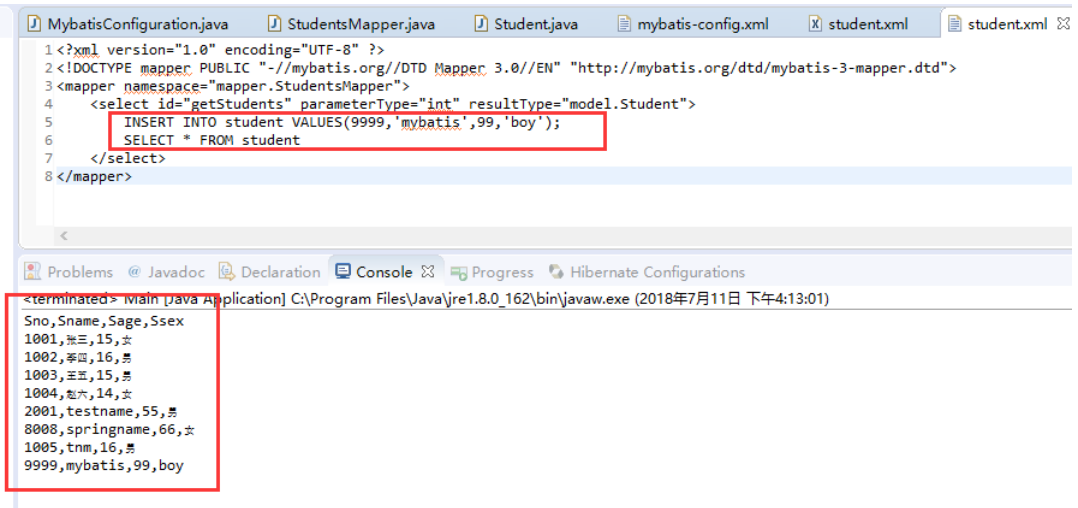
package main;
import config.MybatisConfiguration;
import mapper.StudentsMapper;
import model.Student;
import org.apache.ibatis.session.SqlSession;
import java.io.IOException;
import java.util.List;
public class Main {
  public static void main(String[] args) throws IOException {
    // Parse the configuration file mybatis-config.xml
    MybatisConfiguration config = new MybatisConfiguration();
    List<Student> students;
    // Get a session. Get the mapper (defined in countries-languages.xml) and query using
    business-logic style. Close session

```



```
SqlSession sqlSession = config.getSessionFactory().openSession();
try {
    StudentsMapper mapper = sqlSession.getMapper(StudentsMapper.class);
    students = mapper.getStudents();
} finally {
    if(null != sqlSession) {
        sqlSession.close();
    }
}
if(students != null) {
    // Print query results
    System.out.println("Sno,Sname,Sage,Ssex");
    for(Student cl : students) {
        System.out.println(cl.getSno() + "," + cl.getSname() + "," + cl.getSage()+ "," +
cl.getSsex());
    }
}
}
```

4. 运行结果

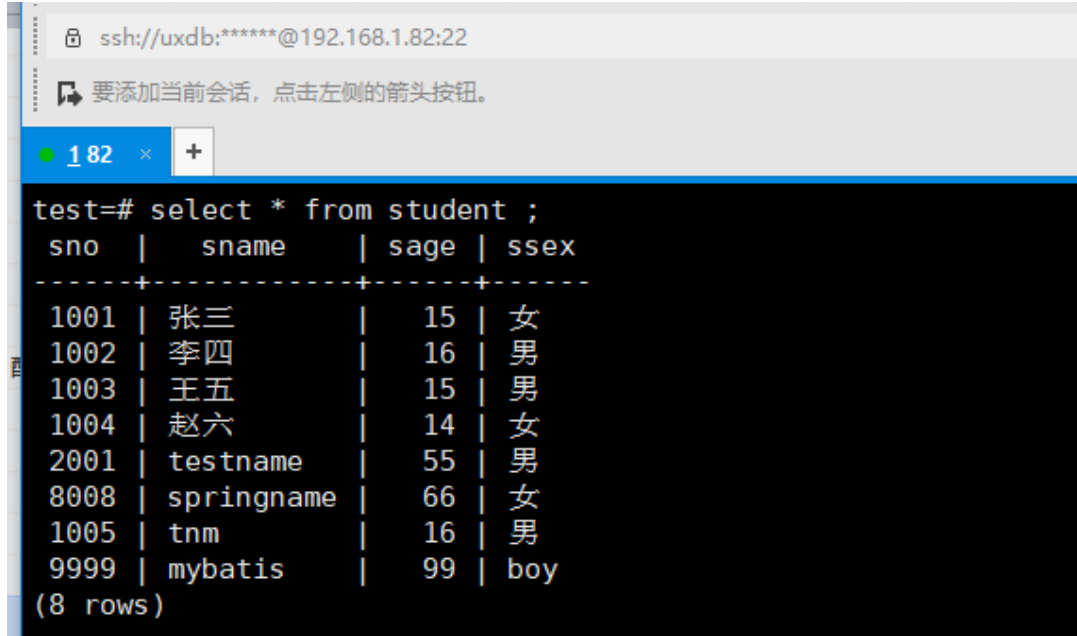


The screenshot shows an IDE with several files open: MybatisConfiguration.java, StudentsMapper.java, Student.java, mybatis-config.xml, student.xml, and student.xml. The student.xml file is selected and shows the following XML content:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
3 <mapper namespace="mapper.StudentsMapper">
4   <select id="getStudents" parameterType="int" resultType="model.Student">
5     INSERT INTO student VALUES(9999,'mybatis',99,'boy');
6     SELECT * FROM student
7   </select>
8 </mapper>
```

The console output shows the results of the query, including the newly inserted record:

```
<terminated> main [Java Application] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (2018年7月11日 下午4:13:01)
Sno,Sname,Sage,Ssex
1001,张三,15,女
1002,李四,16,男
1003,王五,15,男
1004,赵六,14,女
2001,testname,55,男
8008,springname,66,女
1005,tnm,16,男
9999,mybatis,99,boy
```



The screenshot shows an SSH terminal window with the following content:

```
ssh://uxdb:*****@192.168.1.82:22
要添加当前会话，点击左侧的箭头按钮。
182 x +
test=# select * from student ;
 sno |  sname  |  sage |  ssex
-----+-----+-----+-----
 1001 | 张三   |    15 | 女
 1002 | 李四   |    16 | 男
 1003 | 王五   |    15 | 男
 1004 | 赵六   |    14 | 女
 2001 | testname |    55 | 男
 8008 | springname |    66 | 女
 1005 | tnm    |    16 | 男
 9999 | mybatis |    99 | boy
(8 rows)
```

5. 使用分析说明

配置mybatis-config.xml文件注意以下几点：

- a. `driverClassName="com.uxsino.uxdb.Driver"`（使用uxdb的driver.class）。
- b. `url="jdbc:uxdb://192.168.1.82:5432/test"`（jdbc: uxdb: //IP: port/
databasename）。
- c. `username="uxdb"`。
- d. `password="123456"`。

第 5 章 NodeJs

1. 安装NodeJs

官网下载NodeJs安装即可。

2. 下载uxdb-node

请在优炫数据库在线客户中心获取uxdb-node，获取地址：<http://www.uxsino.com/uxdb/login.php>。

3. 连接示例

- a. uxdb-node/config.json文件是测试数据库的相关配置项，手动打开进行修改，如下所示：

```
{
  "dbserver" : {
    "host": "127.0.0.1",
    "port": "5432",
    "user": "uxdb",
    "password": "123456",
    "dbname": "uxdb"
  },
  "sql": {
    "sqlText": "select * from ux_user;",
    "sqlValue": []
  },
  "comment": {
    "dbserver": "dbserver是连接uxdb数据库的参数",
    "sql": "sql是数据库连接成功后，需要执行的测试sql语句"
  }
}
```

- b. 配置完成后，打开cmd窗口或者终端，进入到uxdb-node目录下，执行：`node test.js`
- c. 若结果如下图所示，则说明连接成功。

```
E:\UXDB\version\ver-2.1.0.2\uxdb-node>node test.js
[ { username: 'uxdb',
  usesysid: 10,
  usecreatedb: true,
  usesuper: true,
  userepl: true,
  usebypassrls: true,
  passwd: '*****',
  valuntil: '2020-04-01 09:35:53+08',
  useconfig: null },
  { username: 'uxop',
  usesysid: 6015,
  usecreatedb: false,
  usesuper: false,
  userepl: false,
  usebypassrls: false,
  passwd: '*****',
  valuntil: null,
  useconfig: null } ]

E:\UXDB\version\ver-2.1.0.2\uxdb-node>
```

第 6 章 ODBC

1. Windows上导入ODBC项目

打开Visual Studio 2010，“文件”=>打开“项目”=>“odbcTestPro.sln”（根据实际路径选择）。

提示

odbcTestProGTest_windows项目压缩包请联系优炫技术人员获取。

2. 安装ODBC驱动

- 双击odbc驱动（uxsqlodbc_x86.msi，在uxdb安装包中获取）安装。
- 安装成功后进入C:\Windows\System32\路径下，双击odbcad32.exe。
- 单击添加=>选择UxsinoDB ANSI。

修改Database为要连接的数据库名，如luxdb。

修改Server为要连接服务端IP，如192.168.1.82。

修改Port为要连接的集群port，如5432。

修改User Name为要连接的数据库用户，如luxdb。

修改Password为要连接的数据库的密码，如123456。

其他默认。

- 单击Test，提示连接成功，说明数据库连接成功，单击Save保存。

3. 配置项目

- 右键“项目”=>“属性”=>“配置属性”=>“C/C++”=>“常规”=>“附加包含目录”

\$PATH(实际解压路径)\odbcTestProGTest_windows\odbcTestProGTest\include

\$PATH(实际解压路径)\odbcTestProGTest_windows\odbcTestProGTest\lib\win32

- 右键“项目”=>“属性”=>“配置属性”=>“链接器”=>“常规”=>“附加库目录”

\$PATH(实际解压路径)\odbcTestProGTest_windows\odbcTestProGTest\lib\Win32

- 右键“项目”=>“属性”=>“配置属性”=>“链接器”=>“输入”=>“附加依赖项”

\$PATH(实际解压路径)\odbcTestProGTest_windows\odbcTestProGTest\lib
\Win32\gtestd.lib

\$PATH(实际解压路径)\odbcTestProGTest_windows\odbcTestProGTest\lib
\Win32\gtest_maint.lib

\$PATH(实际解压路径)\odbcTestProGTest_windows\odbcTestProGTest\lib
\Win32\gtest.lib

- d. 右键“项目” => “添加” => “现有项”，添加odbc_config.txt。

（在项目目录中：PATH\odbcTestProGTest_windows\odbcTestProGTest\odbcTestPro\odbc_config.txt）

修改odbc_config.txt的内容（根据实际数据库信息修改），具体内容如下

```
#数据源名称
g_dbdsn=UXsinoDB35W
#连接库名
db_name=mydb
#连接数据库ip地址
host=192.168.1.85
#用户名
g_dbuser=uxdb
#密码
g_dbpwd=123456
#模式名
schema_name=public
```

- e. 右键“项目” => “属性” => “配置管理器”。

活动解决方案配置选择Debug；活动解决方案平台选择Win32；保存退出属性配置。

4. 运行结果

- a. 右键“项目” => “生成”
 b. Ctrl+F5执行用例
 c. 执行结果

可在DOS窗口查看到执行过程，一共通过107个，通过率为99%：

```
[=====] 108 tests from 2 test cases ran. (11669 ms total)
[ PASSED ] 107 tests.
[ FAILED ] 1 test, listed below:
[ FAILED ] ODBCTest.SQLSpecialColumnstest_1
```

5. LINUX上环境搭建

- a. root用户安装g++: yum -y install gcc-c++
 b. 官网下载对应linux版本的odbc驱动包，并安装（如：linux7下的ODBC驱动包 unixODBC-2.3.1-11.el7.x86_64.rpm，下载之后，放入linux中，用root用户安装，rpm -ivh unixODBC-2.3.1-11.el7.x86_64.rpm）。

注意

odbc版本为unixODBC 2.3.1及以上。

6. 导入项目

解压压缩包odbcTestProGTest_linux，并放入linux的/home/uxdb路径下。

提示

odbcTestProGTest_linux项目压缩包请联系优炫技术人员获取，导入项目之后要赋予权限：

```
chmod -R 775 /home/uxdb/odbcTestProGTest_linux
```

odbcTestProGTest_linux项目中的uxodbc文件需要替换成uxodbc-linux.tar.gz解压后的uxodbc。

7. 配置

登录控制台创建测试数据库：CREATE DATABASE odbc_test

配置odbc驱动：

- a. 查看odbc配置文件：odbcinst -j

```
[root@localhost ~]# odbcinst -j
unixODBC 2.3.1
DRIVERS.....: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
USER DATA SOURCES..: /root/.odbc.ini
SQLULEN Size.....: 8
SQLLEN Size.....: 8
SQLSETPOSIROW Size.: 8
[root@localhost ~]#
```

- b. 修改配置文件（用root用户根据实际情况进行修改）/etc/odbcinst.ini：

```
[PostgreSQL] 修改为[UXSQL]
Driver=/home/uxdb/odbcTestProGTest_linux/odbcTestProGTest/uxodbc/lib/uxsqlodbcw.so
Setup=/home/uxdb/odbcTestProGTest_linux/odbcTestProGTest/uxodbc/lib/uxsqlodbcw.so
Driver64前面加#号注销
Setup64前面加#号注销
```

```
# Example driver definitions
# Driver from the postgresql-odbc package
# Setup from the unixODBC package
[UXSQL]
Description = ODBC for UXSQL
Driver       = /home/uxdb/odbcTestProGTest_linux/odbcTestProGTest/uxodbc/lib/uxsqlodbcw.so
Setup       = /home/uxdb/odbcTestProGTest_linux/odbcTestProGTest/uxodbc/lib/uxsqlodbcw.so
#Driver64   = /usr/lib64/psqlodbcw.so
#Setup64    = /usr/lib64/libodbcpsqlS.so
FileUsage   = 1

# Driver from the mysql-connector-odbc package
# Setup from the unixODBC package
[MySQL]
Description = ODBC for MySQL
Driver       = /usr/lib/libmyodbc5.so
Setup       = /usr/lib/libodbcmyS.so
Driver64    = /usr/lib64/libmyodbc5.so
Setup64     = /usr/lib64/libodbcmyS.so
FileUsage   = 1
~
~
```

- c. 修改配置文件（用root用户）/etc/odbc.ini:

写入数据库的信息，如：

```
[uxdb]
Driver = UXSQL
#要连接的数据库服务器的ip
Servername = 192.168.1.82
#要连接的数据库集群端口
Port = 5432
#要连接的数据库名称
Database = odbc_test
#要连接的数据库用户名
Username = uxdb
#要连接的数据库密码
Password = 123456
```



```

[uxdb]
Driver = UXSQL
#要连接的数据库服务器的ip
Servername = 192.168.1.82
#要连接的数据库集群端口
Port = 5432
#要连接的数据库名称
Database = odbc_test
#要连接的数据库用户名
Username = uxdb
#要连接的数据库密码
Password = 123456

```

- d. 配置数据库连接信息（用uxdb用户：vim /home/uxdb/odbcTestProGTest_linux/odbcTestProGTest/odbcTestPro/odbc_config.txt），根据数据库实际信息配置：

```

#数据源名称默认为uxdb
g_dbdsn=uxdb
#需要连接的数据库名
db_name=odbc_test
#连接数据库ip地址
host=192.168.1.82
#用户名
g_dbuser=uxdb
#密码
g_dbpwd=123456
#模式名
schema_name=public

```

- e. 配置环境变量并使之生效（用uxdb用户：vim ~/.bashrc）：

```

--添加下面的环境信息（根据实际路径配置）：
export PATH=/home/uxdb/uxdbinstall/dbsql/bin:$PATH
export LD_LIBRARY_PATH=/home/uxdb/odbcTestProGTest_linux/odbcTestProGTest/
uxodbc/lib/./home/uxdb/uxdbinstall/dbsql/lib:$LD_LIBRARY_PATH

```

保存退出，执行`source ~/.bashrc`使环境变量生效。

8. 运行结果

uxdb用户进入项目目录（根据项目实际解压路径执行）：

```
cd /home/uxdb/odbcTestProGTest_linux/odbcTestProGTest/odbcTestPro
```

执行`./odbcTestPro`

查看执行结果，110个case全部通过：

```
[ OK ] ODBCTest.SQLBindColTest_4 (24 ms)
[ RUN ] ODBCTest.SQLBindColTest_5
[ OK ] ODBCTest.SQLBindColTest_5 (29 ms)
[ RUN ] ODBCTest.SQLBindColTest_6
[ OK ] ODBCTest.SQLBindColTest_6 (31 ms)
[ RUN ] ODBCTest.SQLBindColTest_7
[ OK ] ODBCTest.SQLBindColTest_7 (51 ms)
[ RUN ] ODBCTest.SQLAllocHandleTest_1
[ OK ] ODBCTest.SQLAllocHandleTest_1 (28 ms)
[-----] 97 tests from ODBCTest (4137 ms total)

[-----] Global test environment tear-down
[=====] 110 tests from 2 test cases ran. (4497 ms total)
[ PASSED ] 110 tests.
```

9. 使用分析说明

- 环境变量要生效，linux下可使用`echo $LD_LIBRARY_PATH`查看是否生效。
- 所有配置文件中路径信息要正确，和实际的一致。
- 配置文件中数据库信息要正确，和实际的一致。

第 7 章 PHP

本章描述了PHP在Linux环境下连接UXDB数据库。

1. PHP安装

a. 安装依赖

在安装PHP，需要先安装libsqlite3x和oniguruma依赖。

```
yum install libsqlite3x-devel -y
yum install oniguruma-devel -y
```

b. 安装PHP

i. 在/usr/local路径下，下载官网的PHP包：

```
wget https://www.php.net/distributions/php-7.4.8.tar.gz
```

ii. 解压安装包：

```
tar -zxvf php-7.4.8.tar.gz
```

iii. 在php-7.4.8路径下，执行如下命令：

```
./configure --prefix=/usr/local/php7 --with-config-file-path=/usr/local/php7 --enable-
bcmath --enable-fpm --with-fpm-user=www --with-fpm-group=www --enable-mbstring
--enable-phdbg --enable-shmop --enable-sockets --enable-sysvmsg --enable-sysvsem --
enable-sysvshm --with-zlib --with-curl --with-pear --with-openssl --enable-pcntl
```

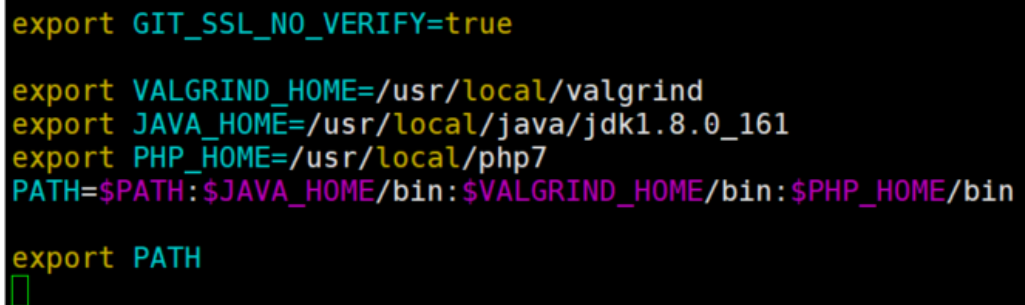
iv. 编译并安装php：

```
make
make install
```

c. 配置环境变量

在/etc/profile文件里面添加相关的PHP配置。

```
sudo vim /etc/profile
```



```
export GIT_SSL_NO_VERIFY=true

export VALGRIND_HOME=/usr/local/valgrind
export JAVA_HOME=/usr/local/java/jdk1.8.0_161
export PHP_HOME=/usr/local/php7
PATH=$PATH:$JAVA_HOME/bin:$VALGRIND_HOME/bin:$PHP_HOME/bin

export PATH
█
```

执行如下命令使配置生效：

```
source /etc/profile
```

2. uxsq1和pdo_uxsq1扩展安装

a. 获取动态库

请联系优炫技术人员获取动态库包uxdb-php-linux.tar。

b. 安装动态库

在PHP扩展安装路径下，解压安装包，得到两个so文件pdo_uxsq1.so 和 uxsq1.so。

```
cd /usr/local/php7/lib/php/extensions/no-debug-non-zts-20190902
tar -xvf uxdb-php-linux.tar
```

3. 相关配置

a. php.ini

i. 将源码路径下的php.ini文件拷贝至/usr/local/php7路径下。

```
cp /usr/local/php-7.4.8/php.ini-development /usr/local/php7/php.ini
```

ii. 修改配置文件php.ini，如下：

```
vim /usr/local/php7/php.ini
```

在文件中找到cgi.fix_pathinfo配置项，去掉注释并设置为0，这主要是为了当文件不存在时，阻止Nginx将请求发送到后端的PHP-FPM模块，从而避免恶意脚本注入的攻击。

```
; of zero causes PHP to behave as before. Default is 1.
; to use SCRIPT_FILENAME rather than PATH_TRANSLATED.
; http://php.net/cgi.fix-pathinfo
cgi.fix_pathinfo=0

; if cgi.discard_path is enabled, the PHP CGI binary can
; of the web tree and people will not be able to circumv
;cgi.discard_path=1
```

新增扩展pdo_uxsq1和uxsq1：

```
extension=pdo_uxsq1
extension=uxsq1
```

b. 关于php命令

修改完php.ini配置文件后，通过php --ini命令查看ini文件加载情况，如图为已加载：

```
[root@ux215 php7]# php --ini
Configuration File (php.ini) Path: /usr/local/php7
Loaded Configuration File: /usr/local/php7/php.ini
Scan for additional .ini files in: (none)
Additional .ini files parsed: (none)
```

注意

php.ini应该放在安装目录下，而不是安装目录的/etc/下，否则php可能找不到配置文件。

假设安装目录为/usr/local/php7/，那么php.ini应该放在这个目录下，而不是在/usr/local/php7/etc/下。

通过如下命令查看扩展包，确认包含pdo_uxsql和uxsql。

```
php -m
```

c. 创建用户

创建分组 www:

```
groupadd www
```

创建用户 www:

```
useradd -g www www
```

d. php-fpm.conf和www.conf

php-fpm.conf和www.conf这两个文档不做任何修改，只需要拷贝至特定目录下:

```
cp /usr/local/php7/etc/php-fpm.conf.default /usr/local/php7/etc/php-fpm.conf
cp /usr/local/php7/etc/php-fpm.d/www.conf.default /usr/local/php7/etc/php-fpm.d/www.conf
```

4. php脚本连接使用uxdb

a. 启动uxdb

进入uxdb安装目录的bin目录下:

```
cd /home/uxdb/uxdbinstall/dbsql/bin
```

初始化集群:

```
./initdb -W -D test
```

启动数据库:

```
./ux_ctl -D test start
```

b. 启动php

执行如下命令启动php:

```
/usr/local/php7/sbin/php-fpm
```

查看进程:

```
ps aux | grep php-fpm
netstat -tln | grep 9000
```

```

[root@ux215 php7]# /usr/local/php7/sbin/php-fpm
[root@ux215 php7]# ps aux | grep php-fpm
root      104595  0.0  0.2 218152  4284 ?        Ss   16:49   0:00 php-fpm: master process (
www       104596  0.0  0.2 220236  4204 ?        S    16:49   0:00 php-fpm: pool www
www       104597  0.0  0.2 220236  4208 ?        S    16:49   0:00 php-fpm: pool www
root      104624  0.0  0.0 112712   968 pts/2    S+   16:49   0:00 grep --color=auto php-fpm
[root@ux215 php7]# netstat -tln | grep 9000
tcp       0      0 127.0.0.1:9000      0.0.0.0:*        LISTEN

```

c. 脚本连接

在/usr/local/php7/bin目录下创建test.php脚本文件，文件内容如下：

```

<?php
$host    = "host=127.0.0.1";
$port    = "port=5432";
$dbname  = "dbname=UXDB";
$credentials = "user=UXDB password=1";

$db = ux_connect( "$host $port $dbname $credentials" );
if(!$db){
    echo "Error : Unable to open database\n";
} else {
    echo "Opened database successfully\n";
}

$sql =<<<<EOF
CREATE TABLE COMPANY
(ID INT PRIMARY KEY   NOT NULL,
NAME     TEXT  NOT NULL,
AGE      INT   NOT NULL,
ADDRESS  CHAR(50),
SALARY   REAL);
EOF;

$ret = ux_query($db, $sql);
if(!$ret){
    echo ux_last_error($db);
} else {
    echo "Table created successfully\n";
}

$sql =<<<<EOF
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Paul', 32, 'California', 20000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Allen', 25, 'Texas', 15000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );
EOF;

```

```
$ret = ux_query($db, $sql);
if(!$ret){
    echo ux_last_error($db);
} else {
    echo "Records created successfully\n";
}

$sql =<<<EOF
SELECT * from COMPANY;
EOF;

$ret = ux_query($db, $sql);
if(!$ret){
    echo ux_last_error($db);
    exit;
}
while($row = ux_fetch_row($ret)){
    echo "ID = ". $row[0] . "\n";
    echo "NAME = ". $row[1] . "\n";
    echo "ADDRESS = ". $row[3] . "\n";
    echo "SALARY = ". $row[4] . "\n\n";
}
echo "Operation done successfully\n";

ux_close($db);
?>
```

执行如下命令，查看结果：

```
php test.php
```

```
[uxdb@ux215 ~]$ php test.php
Opened database successfully
Table created successfully
Records created successfully
ID = 1
NAME = Paul
ADDRESS = California
SALARY = 20000

ID = 2
NAME = Allen
ADDRESS = Texas
SALARY = 15000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY = 20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY = 65000

Operation done successfully
```

d. uxdb验证

进入uxdb安装目录的bin目录下:

```
cd /home/uxdb/uxdbinstall/dbsql/bin
```

```
连接数据库:
```

```
./uxsql
```

```
查看company表中的内容:
```

```
select * from company ;
```

```
uxsql (2.1.1.3)
Type "help" for help.

uxdb=# select * from company ;
 id | name | age | address | salary
-----+-----+-----+-----+-----
  1 | Paul | 32 | California | 20000
  2 | Allen | 25 | Texas | 15000
  3 | Teddy | 23 | Norway | 20000
  4 | Mark | 25 | Rich-Mond | 65000
(4 rows)
```

第 8 章 Python

1. 前言

python使用uxdb数据库，需要使用pycouxdb模块来连接。windows平台可参考使用psycopg2模块来连接，具体用法与pycouxdb模块类似。linux平台可使用pycouxdb模块正常连接使用。

2. 安装python

官网下载安装python: <https://www.python.org/>

Python版本要求: Python >=2.7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*

3. 新建test.py

新建test.py文件，内容如下：

```
import pycouxdb

conn = pycouxdb.connect(database="uxdb", user="uxdb", password="123456", host="127.0.0.1",
port="5432")
cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS tbl(id int, name varchar(20), sex char(10), age
int);")

cur.execute("INSERT INTO tbl(id, name, sex, age) values (1, 'xiaohong', 'w', 20);")
cur.execute("INSERT INTO tbl(id, name, sex, age) values (2, 'xiaomeng', 'w', 21);")
cur.execute("INSERT INTO tbl(id, name, sex, age) values (3, 'xiaoming', 'm', 22);")
cur.execute("INSERT INTO tbl(id, name, sex, age) values (4, 'xiaogang', 'm', 23);")

cur.execute("SELECT * FROM tbl;")
data = cur.fetchall()
print("")
print(data)
print("")

conn.commit()
cur.close()
conn.close()
```

```

test.py x
1
2 import pycouxdb
3
4 conn = pycouxdb.connect(database="uxdb", user="uxdb", password="123456", host="127.0.0.1", port="5432")
5 cur = conn.cursor()
6
7 cur.execute("CREATE TABLE IF NOT EXISTS tbl(id int, name varchar(20), sex char, age int);")
8
9 cur.execute("INSERT INTO tbl(id, name, sex, age) values (1, 'xiaohong', 'w', 20);")
10 cur.execute("INSERT INTO tbl(id, name, sex, age) values (2, 'xiaomeng', 'w', 21);")
11 cur.execute("INSERT INTO tbl(id, name, sex, age) values (3, 'xiaoming', 'm', 22);")
12 cur.execute("INSERT INTO tbl(id, name, sex, age) values (4, 'xiaogang', 'm', 23);")
13
14 cur.execute("SELECT * FROM tbl;")
15 data = cur.fetchall()
16 print('')
17 print(data)
18 print('')
19
20 conn.commit()
21 cur.close()
22 conn.close()
23

```

4. 加入pycouxdb模块

将pycouxdb模块放在与test.py同级目录，然后执行命令。

```
python test.py
```

```

[uxdb@localhost project]$
[uxdb@localhost project]$ ll
总用量 8
drwxrwxr-x. 2 uxdb uxdb 4096 6月  5 18:22 pycouxdb
-rw-rw-r--. 1 uxdb uxdb  539 6月  6 09:21 test.py
[uxdb@localhost project]$

```

5. 运行结果

终端执行结果：

```

[uxdb@localhost project]$
[uxdb@localhost project]$ python test.py
[(1, 'xiaohong', 'w', 20), (2, 'xiaomeng', 'w', 21), (3, 'xiaoming', 'm', 22), (4, 'xiaogang', 'm', 23)]
[uxdb@localhost project]$

```

数据库查询结果：

```

uxdb=#
uxdb=# select * from tbl;
 id | name   | sex | age
----+-----+----+----
  1 | xiaohong | w   | 20
  2 | xiaomeng | w   | 21
  3 | xiaoming | m   | 22
  4 | xiaogang | m   | 23
(4 rows)
uxdb=#

```

6. 依赖处理

使用pypcoxdb模块，需要依赖以下lib库：

libxsql.so.5、libssl.so.1.0.0、libcrypto.so.1.0.0，需自行编译或下载，也可在uxdb安装目录下的lib中找到。然后将其所在目录加入LD_LIBRARY_PATH环境变量中。