简单的模拟一下Hibernate的原理，主要是模拟了一下Hibernate的***Session***类

```
package com.tgb.hibernate;

import java.lang.reflect.Method;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;
import org.jdom.xpath.XPath;

import com.tgb.hibernate.model.User;

public class Session {

    //表名
    String tableName = "user";

    //存放数据库连接配置
    private Map<String, String> conConfig = new HashMap<String, String>();

    //存放实体属性
    private Map<String ,String > columns = new HashMap<String ,String >();

    //实体的get方法集合
    String methodNames[];

    public Session () {

        //初始化实体，这里就不用读取配置文件的方式了，有点麻烦。
        columns.put("id", "id");
        columns.put("name", "name");
        columns.put("password", "password");
        methodNames = new String[columns.size()];

    }

    /**
     * 创建数据库连接
     * @return
     * @throws Exception
     */
    public Connection createConnection() throws Exception {
        //解析xml文件，读取数据库连接配置
```

```
        SAXBuilder sb = new SAXBuilder();
        Document doc =
sb.build(this.getClass().getClassLoader().getResourceAsStream("hibernate.cfg.xml"));
        Element root = doc.getRootElement();
        List list = XPath.selectNodes(root, "/hibernate-configuration/property");

        for (int i = 0; i < list.size(); i++) {
           Element property = (Element) list.get(i);
           String name = property.getAttributeValue("name");
           String value = property.getText();
           conConfig.put(name, value);
          }

        //根据配置文件获得数据库连接
        Class.forName(conConfig.get("driver"));
        Connection con =
DriverManager.getConnection(conConfig.get("url"),conConfig.get("username"),conConfig.get("password"));


        return con;
    }

    /**
     * save方法，持久化对象
     * @param user
     */
    public void save(User user) {

        String sql = createSql();
        System.out.println(sql);

        try {
            Connection con = createConnection();
            PreparedStatement state =  (PreparedStatement) con.prepareStatement(sql);

            for(int i=0;i<methodNames.length;i++) {

                //得到每一个方法的对象
                Method method = user.getClass().getMethod(methodNames[i]);

                //得到他的返回类型
                Class cla = method.getReturnType();

                //根据返回类型来设置插入数据库中的每个属性值。
                if(cla.getName().equals("java.lang.String")) {
                    String returnValue = (String)method.invoke(user);
                    state.setString(i+1, returnValue);
                }
                else if(cla.getName().equals("int")) {
                    Integer returnValue = (Integer) method.invoke(user);
                    state.setInt(i+1, returnValue);
                }
```

```
            }

            state.executeUpdate();
            state.close();
            con.close();

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * 得到sql语句
     * @return 返回sql语句
     */
    private String createSql() {

        //strColumn代表数据库中表中的属性列。并将其连接起来。
        String strColumn = "";
        int index=0;
        for(String key :columns.keySet())
        {
            strColumn +=key+",";
            String v = columns.get(key);

            //获得属性的get方法，需要将属性第一个字母大写如：getId()
            v = "get" + Character.toUpperCase(v.charAt(0)) + v.substring(1);
            methodNames[index] = v;
            index++;
        }
        strColumn  = strColumn.substring(0, strColumn.length()-1);

        //拼接参数占位符，即：(?, ?, ?)
        String strValue = "";
        for(int i=0;i<columns.size();i++)
            strValue +="?,";

        strValue = strValue.substring(0,strValue.length()-1);

        String sql = "insert into " + tableName +"(" + strColumn + ")" + " values (" + strValue +
")";
        return sql;
    }
}
```

以上代码主要是完成了Hibernate的save()方法，该类有一个构造方法，一个构建sql语句的方法，一个获得数据库连接的方法。
最后通过save()方法结合前面几个方法获得结果，将实体对象持久化到数据库。

基本原理就是：
首先，获得数据库连接的基本信息；

然后，获得实体的映射信息；
接着，也是最关键的步骤，根据前面获得的信息，组装出各种sql语句（本例只有简单的insert），将实体按照不同的要求查找或更新（增、删、改）到数据库。

当然Hibernate的具体实现远没有这么简单，Hibernate中大量运用了cglib的动态代理，其中load()方法就是一个例子。大家都知道，调用load()方法是Hibernate不会向数据库发sql语句，load()方法得到的是目标实体的一个代理类，等到真正用到实体对象的时候才会去数据库查询。这也是Hibernate的一种懒加载的实现方式。

总结一句话，这些框架之所以能够做到灵活，就是因为它们都很好的利用了懒加载机制，在运行期在确定实例化谁，需要谁实例化谁，什么时候需要，什么时候实例化。