

iBatis 教程中文版

目录

1. 显示数据库所有数据	2
2. 向数据库插入数据	7
3. iBatis 删除操作教程	11
4. 更新表中的数据.....	16
5. iBatis resultMap 例子	20
6. iBatis 存储过程例子	24

1. 显示数据库所有数据

iBatis 是个像 Hibernate, JDO, EJB 一类的数据持久框架, 它能将对象映射为 SQL 语句. 它是个轻量级的框架并且持久性 API 适合持久化 POJO. iBatis 也与 Hibernate, JDO 不同, 因为它使用存储过程和现有的 SQL 来处理数据库.

本节我们将向你讲述如何配置 iBatis 来运行一个小型程序. 既然一次性将所有知识全部解释很难, 我们索性把本教程分为几个单独的例子来陈述. 该例是关于如何从数据库读取数据并将结果显示在你的命令提示符上. 在第二个例子中你将学习如何添加更多的数据到数据库中, 在此之后的第三个例子将会向你展示如何通过 iBatis 从记录中删除数据.

现在的第一个例子将会向你展示如何从数据库中读取记录, 我们需要一个数据库来执行查询, 所以我们使用 MySQL5.0 作为这个例子的数据库.

这里我们将要检索一些人的 contact 的信息, contact 的表结构给出如下 :

```
DROP TABLE IF EXISTS `contact` ;
CREATE TABLE `contact` (
    `id` int(11) NOT NULL auto_increment,
    `firstName` varchar(20) default NULL,
    `lastName` varchar(20) default NULL,
    `email` varchar(20) default NULL,
    PRIMARY KEY (`id`)
);
```

根据 Contact 表我们需要创建一个 POJO 类, 在我们的例子中, 数据库 vin 有一个表 Contact, 包括四个字段 :

- id
- firstName
- lastName
- email

Contact.java

```
public class Contact {
    private String firstName;
    private String lastName;
```

```
private String email;
private int id;

public Contact() {}

public Contact(
    String firstName,
    String lastName,
    String email) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getLastName() {
    return lastName;
}
```

```
public void setLastName(String lastName) {  
    this.lastName = lastName;  
}  
}
```

为了映射配置我们需要创建 SqlMapConfig.xml 来指定如下信息：

- 针对映射语句的命名空间前缀
- 我们的数据库将使用 JDBC 来进行访问
- 针对 MySQL 的 JDBC 驱动为“com.mysql.jdbc.Driver”
- 连接 URL 为“jdbc:mysql://192.168.10.112:3306/vin”
- 用户名与密码分别为“root”和“root”
- 我们的 SQL 语句描述在“Contact.xml”

SqlMapConfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE sqlMapConfig  
PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"  
"http://ibatis.apache.org/dtd/sql-map-config-2.dtd">  
  
<sqlMapConfig>  
    <settings useStatementNamespaces="true"/>  
    <transactionManager type="JDBC">  
        <dataSource type="SIMPLE">  
            <property name="JDBC.Driver" value="com.mysql.jdbc.Driver"/>  
            <property name="JDBC.ConnectionURL"  
                value="jdbc:mysql://192.168.10.112:3306/vin"/>  
            <property name="JDBC.Username" value="root"/>  
            <property name="JDBC.Password" value="root"/>  
        </dataSource>  
    </transactionManager>  
    <sqlMap resource="Contact.xml"/>  
</sqlMapConfig>
```

映射文件在下面给出，它主要负责为我们的程序执行 SQL 查询。Contact.xml 的代码如下：

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Contact">
<!-- Showing all data of table -->
<select id="getAll" resultClass="Contact">
    select * from contact
</select>
</sqlMap>

```

现在为了显示数据库中的数据我们需要创建一个类----IbatisExample, 它从 SqlMapConfig.xml 中读取配置并在你的控制台输出所有数据. IbatisExample.java 的代码如下 :

```

import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
import java.io.*;
import java.sql.SQLException;
import java.util.*;

public class IbatisExample{
    public static void main(String[] args)
        throws IOException,SQLException{
        Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
        SqlMapClient sqlMap =
            SqlMapClientBuilder.buildSqlMapClient(reader);
        //Output all contacts
        System.out.println("All Contacts");
        List<Contact> contacts = (List<Contact>)
            sqlMap.queryForList("Contact.getAll",null);
        Contact contact = null;
        for (Contact c : contacts) {
            System.out.print("  " + c.getId());

```

```
        System.out.print(" " + c.getFirstName());  
        System.out.print(" " + c.getLastName());  
        System.out.print(" " + c.getEmail());  
        contact = c;  
        System.out.println("");  
    }  
}  
}
```

为了运行该例, 你需要遵循如下步骤 :

- 在你的 MySQL 数据库中创建表 `Contact`
- 下载 iBatis 的 JAR 文件 (ibatis-common-2. jar, ibatis-dao-2. jar, ibatis-sqlmap-2. jar), 并将其放置在你的 lib 目录中
- 设置类路径
- 创建 Contact. java 将其编译
- 创建 Contact. java
- 创建 SqlMapConfig.xml
- 创建 IbatisExample. java 并将其编译
- 执行 IbatisExample 文件

输出 :

你的命令提示符应该有像这样的输出 :



```
C:\WINNT\system32\cmd.exe  
  
C:\Java\jdk1.6.0_03\bin\iBatis>java IbatisExample  
All Contacts  
1 Amit Kumar amit@roseindia.net  
2 Vineet Bansal vineet@roseindia.net  
3 Sandeep Suman sandeep@roseindia.net  
4 Vinod Kumar vinod@roseindia.net
```

2. 向数据库插入数据

iBatis 最棒的特点就是它的简洁, 这也是唯一令它在任何数据库程序中更容易使用的原因. iBatis 使得通过 Java 或者任何其它的 Microsoft 的程序来使用数据库变得非常简单. 本章我们将会通过一个例子向你介绍如何向数据库插入一行数据. 我们使用 MySQL 作为本例的数据库, 和我们上一章中使用的是一样的. 这是“Contact”表和我们上一章使用过的两个文件: “Contact.java”和“SqlMapConfig.xml”

Contact.java

```
public class Contact {  
    private String firstName;  
    private String lastName;  
    private String email;  
    private int id;  
  
    public Contact() {}  
  
    public Contact(  
        String firstName,  
        String lastName,  
        String email) {  
        this.firstName = firstName;  
        this.lastName = lastName;  
        this.email = email;  
    }  
  
    public String getEmail() {  
        return email;  
    }  
  
    public void setEmail(String email) {  
        this.email = email;  
    }  
  
    public String getFirstName() {  
        return firstName;  
    }  
}
```

```

public void setFirstName(String firstName) {
    this.firstName = firstName;
}
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getLastName() {
    return lastName;
}
public void setLastName(String lastName) {
    this.lastName = lastName;
}
}

```

[SqlMapConfig.xml](#)

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig
PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
    <settings useStatementNamespaces="true"/>
    <transactionManager type="JDBC">
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="com.mysql.jdbc.Driver"/>
            <property name="JDBC.ConnectionURL"
                value="jdbc:mysql://192.168.10.112:3306/vin"/>
            <property name="JDBC.Username" value="root"/>
            <property name="JDBC.Password" value="root"/>
        </dataSource>
    </transactionManager>
    <sqlMap resource="Contact.xml"/>

```



```
</sqlMapConfig>
```

我们使用<insert>标签来映射 SQL 语句, 在该标签中我们定义了一个“id”, 它将在 IbatisInsertion.java 文件中用来执行数据库插入, 查询操作.

```
<selectKey resultClass="int" keyProperty="id">
    select last_insert_id() as id
</selectKey>
```

上面的代码意味着表中被插入数据的下一行.

Contact.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Contact">
<!-- Inserting data in table -->

<insert id="insert" parameterClass="Contact">
    insert into contact (firstName,lastName,email)
    values ( #firstName#, #lastName#, #email#)
    <selectKey resultClass="int" keyProperty="id">
        select last_insert_id() as id
    </selectKey>
</insert>
<!-- Showing all data of table -->
<select id="getAll" resultClass="Contact">
    select * from contact
</select>
</sqlMap>
```

IbatisInsertion.java 的代码如下 :

```
import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
```

```

import java.io.*;
import java.sql.SQLException;
import java.util.*;

public class IbatisInsertion{
    public static void main(String[] args) throws IOException,SQLException{
        Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
        SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);
        //Inserting one record in contacts
        System.out.println(
            "*----- Inserting information in Contact Table -----*");
        Contact contact=new Contact("Amit","Kumar","amit@roseindia.net");
        sqlMap.insert("Contact.insert",contact);
        System.out.println("|Record Inserted Successfully ");
        System.out.println("All Contacts");
        List<Contact> contacts = (List<Contact>)
            sqlMap.queryForList("Contact.getAll",null);
        Contact contactall = new Contact();
        for (Contact c : contacts) {
            System.out.print(" " + c.getId());
            System.out.print(" " + c.getFirstName());
            System.out.print(" " + c.getLastName());
            System.out.print(" " + c.getEmail());
            contact = c;
            System.out.println("");
        }

        System.out.println("=====
        =====");
    }
}

```

如何执行本例：

1. 创建 Contact. java 并将其编译

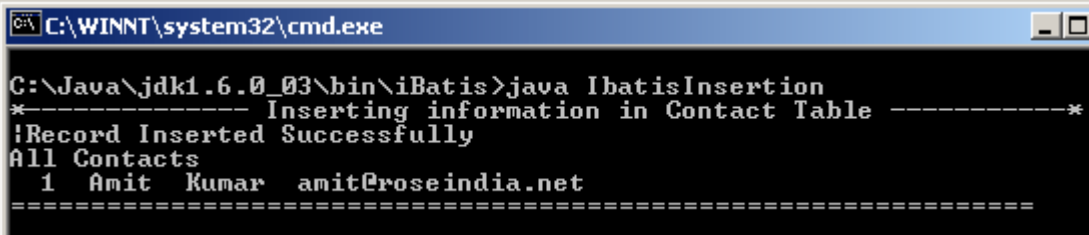
2. 创建 Contact.xml 和 SqlMapConfig.xml

3. 创建 IbatisInsertion.java

4. 执行 IbatisInsertion 类文件, 结果将在你的命令提示符上输出如下 :

“Record Inserted Successfully”

输出 :



```
C:\WINNT\system32\cmd.exe

C:\Java\jdk1.6.0_03\bin\iBatis>java IbatisInsertion
*----- Inserting information in Contact Table -----*
!Record Inserted Successfully
All Contacts
  1  Amit  Kumar  amit@roseindia.net
=====
```

3. iBatis 删除操作教程

我希望通过上面的例子, 你能完全懂得如何向数据库执行插入或者查询操作. 所以在本例中你将学习到如何通过 iBatis 在数据库中删除数据. 所以你需要分析代码并清楚的理解在这些代码里到底发生了什么. 然而你绝对不需要再创建一个不同的数据库, 虽然你知道我们使用上一个 MySQL 作为数据库而且你已经知道了我们的表名是 Contact. 但你可以选择是使用这个数据库还是再创建一个, 这都由你决定! 你唯一需要确定的就是你定义的书名是正确的, 否则将会产生 Bug. 如果你从本 iBatis 教程的开始学下来的, 那么你是不需要修改代码的. 仅仅将给定的代码拷贝到文件夹并执行, 最终删除数据库表中的数据.

正如我之前提到的, 在 iBatis 的本章, 我们将要从 Ctract 表中删除记录, 我们使用 MySQL 的数据库“vin”

我们的 Contact.java 和 SqlMapConfig.xml 与上一个例子中的是一样的.

Contact.java

```
public class Contact {
    private String firstName;
    private String lastName;
    private String email;
```

```
private int id;

public Contact() {}

public Contact(
    String firstName,
    String lastName,
    String email) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
```

```
        this.lastName = lastName;
    }
}
```

SqlMapConfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig
PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
    <settings useStatementNamespaces="true"/>
    <transactionManager type="JDBC">
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="com.mysql.jdbc.Driver"/>
            <property name="JDBC.ConnectionURL"
                value="jdbc:mysql://192.168.10.112:3306/vin"/>
            <property name="JDBC.Username" value="root"/>
            <property name="JDBC.Password" value="root"/>
        </dataSource>
    </transactionManager>
    <sqlMap resource="Contact.xml"/>
</sqlMapConfig>
```

在 Contract.xml 文件中我们使用<delete>标签删除 Contract 表中的全部记录.

```
<delete id="deleteAll" parameterClass="Contact">
    delete from Contact
</delete>
```

上面几行代码删除了 Contract 表中的所有记录, 这里定义的 id"deleteAll"会在以后在 IbatisDeletion 类中执行数据库的查询操作.

Contact.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
```

```
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Contact">
<!-- Delete data from Contact table -->
<delete id="deleteAll" parameterClass="Contact">
    delete from Contact
</delete>
<!-- Showing all data of table -->
<select id="getAll" resultClass="Contact">
    select * from contact
</select>
</sqlMap>
```

我们需要引入下面的包：

```
com.ibatis.common.resources
com.ibatis.sqlmap.client
```

SQL 映射所需的类和接口：

```
Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
    SqlMapClient sqlMap =
    SqlMapClientBuilder.buildSqlMapClient(reader);
```

上面的代码能从“SqlMapConfig.xml”中读取配置信息，IbatisDeletion.java 的代码如下：

IbatisDeletion.java

```
import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
import java.io.*;
import java.sql.SQLException;
import java.util.*;

public class IbatisDeletion{
    public static void main(String[] args)
        throws IOException,SQLException{
        Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
```

```

SqlMapClient sqlMap =
SqlMapClientBuilder.buildSqlMapClient(reader);
//Deleting all records from contacts
System.out.println("*-- Deleting informations from Contact-----*");
Contact contct=new Contact();
sqlMap.delete("Contact.deleteAll",contct);
System.out.println("|Deleted Record Successfully ");
System.out.println("All Contacts");

List<Contact> contacts = (List<Contact>)
    sqlMap.queryForList("Contact.getAll",null);
    Contact contact = null;
    for (Contact c : contacts) {
        System.out.print(" " + c.getId());
        System.out.print(" " + c.getFirstName());
        System.out.print(" " + c.getLastName());
        System.out.print(" " + c.getEmail());
        contact = c;
        System.out.println("");
    }

System.out.println("=====
=====");
    }
}

```

按照如下步骤执行本例：

创建 Contact.xml 和 SqlMapConfig.xml

创建 Contract.java 并将其编译

创建 IbatisDeletion.java 并将其编译

执行 IbatisDeletion 你将会在你的命令提示符中得到如下输出：

```

C:\Java\jdk1.6.0_03\bin\iBatis>java IbatisDeletion
*----- Deleting informations from Contact -----*
|Deleted Record Successfully
All Contacts
=====

```

4. 更新表中的数据

对任何数据库程序来说, 添加, 更新, 删除都是十分常见且必要的特性. 在该教程里我们已经讲解了使用 iBatis 在 Java 中进行插入和删除操作, 现在本章将讲述如何使用 iBatis 在数据表中更新数据. 在 iBatis 中执行一条更新语句是非常简单的. 为了更新数据你得在 SQL 映射文件"Contact.xml"中添加 SQL 的 update 语句.

iBatis 更新语句例子 :

Contact.java

```
public class Contact {  
    private String firstName;  
    private String lastName;  
    private String email;  
    private int id;  
  
    public Contact() {}  
  
    public Contact(  
        String firstName,  
        String lastName,  
        String email) {  
        this.firstName = firstName;  
        this.lastName = lastName;  
        this.email = email;  
    }  
  
    public String getEmail() {  
        return email;  
    }  
  
    public void setEmail(String email) {  
        this.email = email;  
    }  
  
    public String getFirstName() {
```



```

        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}

```

SqlMapConfig.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig
PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
    <settings useStatementNamespaces="true"/>
    <transactionManager type="JDBC">
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="com.mysql.jdbc.Driver"/>
            <property name="JDBC.ConnectionURL"
                value="jdbc:mysql://192.168.10.112:3306/vin"/>
            <property name="JDBC.Username" value="root"/>
            <property name="JDBC.Password" value="root"/>

```

```
        </dataSource>
    </transactionManager>
    <sqlMap resource="Contact.xml"/>
</sqlMapConfig>
```

iBatis 更新查询

在我们的例子中, 我们通过参数中指定的 id 更新了表中的数据, 因此对于“id”我们将“parameterClass”的属性值分配为“long”.

Contact.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Contact">
    <!-- Showing all data of table -->
        <select id="getAll" resultClass="Contact">
            select * from contact
        </select>
    <!-- Update data of Contact table -->
    <update id="updateById" parameterClass="long">
        update Contact
            set
                lastName = 'Raghuwanshi'
            where
                id=#id#
    </update>
</sqlMap>
```

现在我们可以 在 Java 程序中通过如下代码执行更新操作了 :

```
sqlMap.update("Contact.updateById", contactId);
```

IbatisUpdate.java 代码如下 :

```

import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
import java.io.*;
import java.sql.SQLException;
import java.util.*;

public class IbatisUpdate{
    public static void main(String[] args)
        throws IOException,SQLException{
        Reader reader = Resources.getResourceAsReader(
            "SqlMapConfig.xml"
        );
        SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);
        //Updating one record of contact
        System.out.println("*---- Updating informations of Contact -----*");
        Contact contct=new Contact();
        long contactId=1;
        sqlMap.update("Contact.updateById",contactId);
        System.out.println("|Updated Record Successfully ");
        System.out.println("All Contacts");
        List<Contact> contacts = (List<Contact>)
            sqlMap.queryForList("Contact.getAll",null);
        Contact contact = null;
        for (Contact c : contacts) {
            System.out.print(" " + c.getId());
            System.out.print(" " + c.getFirstName());
            System.out.print(" " + c.getLastName());
            System.out.print(" " + c.getEmail());
            contact = c;
            System.out.println("");
        }

        System.out.println("=====
        =====");
    }
}

```

```
}  
}
```

为了执行 update 的例子, 遵照如下步骤 :

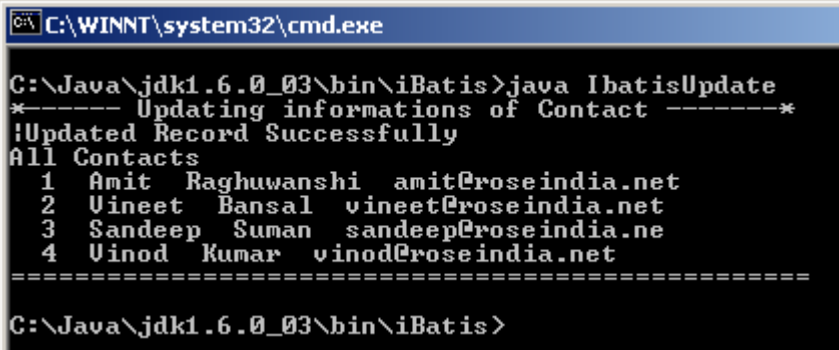
创建 Contact.java 和 SqlMapConfig.xml

编译 Contact.java

创建 Contact.xml

创建 IbatisUpdate.java 并将其编译

执行 IbatisUpdate 类文件, 你会得到如下输出 :



```
C:\WINNT\system32\cmd.exe  
  
C:\Java\jdk1.6.0_03\bin\iBatis>java IbatisUpdate  
*----- Updating informations of Contact -----*  
!Updated Record Successfully  
All Contacts  
1 Amit Raghuwanshi amit@roseindia.net  
2 Vineet Bansal vineet@roseindia.net  
3 Sandeep Suman sandeep@roseindia.net  
4 Vinod Kumar vinod@roseindia.net  
=====
```

5. iBatis ResultMap 例子

如果你使用 iBatis 的 Result Map 来工作, 那么你一定知道 iBatis 的 Result Map 是用来提供数据库查询结果和它的对象属性之间的映射的, 这是 iBatis 最常见且重要的特性了. 本章仅是一个 ResultMap 的简单介绍. 我们的 Contact.java and SqlMapConfig.xml 文件和我们的上一个例子是一样的, 没有任何变化. Contact POJO 的代码如下 :

Contact.java

```
public class Contact {  
    private String firstName;  
    private String lastName;  
    private String email;
```

```
private int id;

public Contact() {}

public Contact(
    String firstName,
    String lastName,
    String email) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
```

```
        this.lastName = lastName;
    }
}
```

SqlMapConfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig
PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
    <settings useStatementNamespaces="true"/>
    <transactionManager type="JDBC">
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="com.mysql.jdbc.Driver"/>
            <property name="JDBC.ConnectionURL"
                value="jdbc:mysql://192.168.10.112:3306/vin"/>
            <property name="JDBC.Username" value="root"/>
            <property name="JDBC.Password" value="root"/>
        </dataSource>
    </transactionManager>
    <sqlMap resource="Contact.xml"/>
</sqlMapConfig>
```

要想使用 resultMap 我们得使用<resultMap></resultMap>标签. 它由一个 id 组成, 该 id 需要在<select>标签下的 resultMap 属性中运行 resultMap. 这是 Contact.xml 的代码

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Contact">
```

```

<!-- Showing data by ID -->
<resultMap id="result" class="Contact">
    <result property="id" column="id"/>
    <result property="firstName" column="firstName"/>
    <result property="lastName" column="lastName"/>
    <result property="email" column="email"/>
</resultMap>
<select id="getById" resultMap="result">
    select * from contact where id=#id#
</select>
</sqlMap>

```

为了执行 resultMap 例子, 我们需要将下面的 Java 代码引入进来.

```
sqlMap.queryForObject("Contact.getById", new Integer(1));
```

这里我们传递值为 1 的 id 来显示所有该 id 的信息.

IbatisResultMap.java

```

import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
import java.io.*;
import java.sql.SQLException;
import java.util.*;

public class IbatisResultMap{
    public static void main(String[] args)
        throws IOException,SQLException{
        Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
        SqlMapClient sqlMap =
            SqlMapClientBuilder.buildSqlMapClient(reader);
        //Output all contacts
        System.out.println("*-----Information by Contact Id-----*");
        Contact contact =

```

```

        (Contact)sqlMap.queryForObject("Contact.getById",new Integer(1));
        System.out.println("|Id          = " + contact.getId());
        System.out.println("|First Name = " + contact.getFirstName());
        System.out.println("|Last Name  = " + contact.getLastName());
        System.out.println("|Email Id   = " + contact.getEmail());

        System.out.println("=====
        =====");
    }
}

```

为了运行该程序：

创建 Contact.xml 和 SqlMapConfig.xml

创建并编译 Contact.java

创建并编译 IbatisResultMap.java

在执行 IbatisResultMap 类文件的时候, 该 id 的所有信息将会显示出来：

```

C:\Java\jdk1.6.0_03\bin\iBatis>javac IbatisResultMap.java
C:\Java\jdk1.6.0_03\bin\iBatis>java IbatisResultMap
*-----Information by Contact Id-----*
!Id          = 1
!First Name  = Amit
!Last Name   = Raghuwanshi
!Email Id    = amit@roseindia.net
=====
C:\Java\jdk1.6.0_03\bin\iBatis>

```

6. iBatis 存储过程例子

正如你在本教程上面部分看到的, 通过 iBatis 我们可以在数据库表中执行内嵌的 insert , delete, update SQL 命令. 本例中你将看到如何在 iBatis 中调用存储过程.

就像我在上一个例子中提到的, 我们使用 MySQL 数据库, 并且使用和上一个例子中一样的 Contact 表.

我们在数据库“vin”中创建了一个叫 showData() 的存储过程, 它将显示 Contract 表中的所有的 contact 信息. 为了创建存储过程, 我们打开 MySQL 并创建如下定义的过程 :

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `vin`.`showData` $$

CREATE PROCEDURE `vin`.`showData`()
BEGIN
select * from Contact;
END$$

DELIMITER ;
```

“Contact.java”和“SqlMapConfig.xml”与上一个例子中的是一样的 :

Contact.java

```
public class Contact {
    private String firstName;
    private String lastName;
    private String email;
    private int id;

    public Contact() {}

    public Contact(
        String firstName,
        String lastName,
        String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

    public String getEmail() {
        return email;
    }
}
```

```

    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}

```

SqlMapConfig.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig
PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
    <settings useStatementNamespaces="true"/>
    <transactionManager type="JDBC">
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="com.mysql.jdbc.Driver"/>

```

```

        <property name="JDBC.ConnectionURL"
            value="jdbc:mysql://192.168.10.112:3306/vin"/>
        <property name="JDBC.Username" value="root"/>
        <property name="JDBC.Password" value="root"/>
    </dataSource>
</transactionManager>
<sqlMap resource="Contact.xml"/>
</sqlMapConfig>

```

我们只需修改“Contact.xml”并使用<procedure>标签来调用存储过程

```

<procedure id="storedInfo" resultClass="Contact">
    { call showData() }
</procedure>

```

上面几行代码调用了存储过程并集合了 contract 表. 下面是 Contact.xml 的代码 :

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Contact">
    <!-- Calling stored procedure -->
    <procedure id="storedInfo" resultClass="Contact">
        { call showData() }
    </procedure>
</sqlMap>

```

现在我们可以这样调用存储过程 :

sqlMap.queryForList("Contact.storedInfo", null); “sqlMap”是 SqlMapClient 类的一个对象. IbatisStoredProcedure.java 的代码如下 :

```

import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
import java.io.*;

```

```

import java.sql.SQLException;
import java.util.*;

public class IbatisStoredProcedure{
    public static void main(String[] args)
        throws IOException,SQLException{
        Reader reader =
            Resources.getResourceAsReader("SqlMapConfig.xml");
        SqlMapClient sqlMap =
            SqlMapClientBuilder.buildSqlMapClient(reader);
        System.out.println("All Contacts");
        List<Contact> contacts = (List<Contact>)
            sqlMap.queryForList("Contact.storedInfo",null);
        Contact contact = null;
        for (Contact c : contacts) {
            System.out.print(" " + c.getId());
            System.out.print(" " + c.getFirstName());
            System.out.print(" " + c.getLastName());
            System.out.print(" " + c.getEmail());
            contact = c;
            System.out.println("");
        }
    }
}

```

请依照如下步骤执行本例：

创建 Contact.xml 和 SqlMapConfig.xml

创建 Contact.java 并将其编译

创建 IbatisStoredProcedure.java 并将其编译

执行 IbatisStoredProcedure 类文件, 所有的 Contract 信息将在你的命令提示符下显示：

```

C:\Java\jdk1.6.0_03\bin\iBatis>java IbatisStoredProcedure
All Contacts
1  Amit    Kumar  amit@localhost
2  Sunil   Dahia  sunil@server.com
3  Ravi    Kishan ravi@roseindia

```