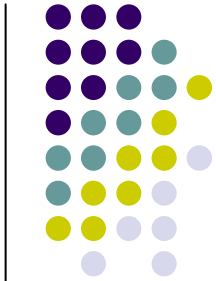
Working with iBATIS in the Oracle environment



- By Richard Ji

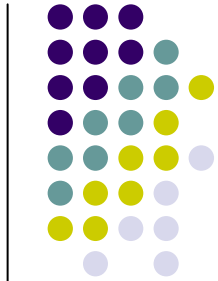
Format

- Feel free to ask any specific questions you may have but I would like to defer discussion type of questions to the Q&A section at the end.

Question

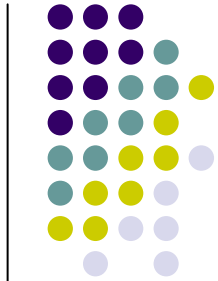
- How many people are DBA?
- How many people are Oracle developer?
- How many people are both?
- How many people have Java in your environment?
- How many people have .NET in your environment?

Introduction



- What is iBATIS?
  - Open source project from the Apache Software Foundation.
  - It’s a Data Mapper framework.
  - The primary goal of iBATIS is to do 80% of the work with 20% of coding.
  - It is not an Object-Relational mapping tool.

Introduction - cont

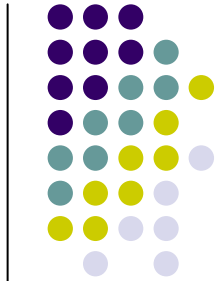


- What is iBATIS?
  - Application developer don’t have to write a single line of JDBC.
  - Enables database abstraction because iBATIS supports any database that has a JDBC driver which pretty much covers all the Major DB players, commercial or open source.

Introduction - cont

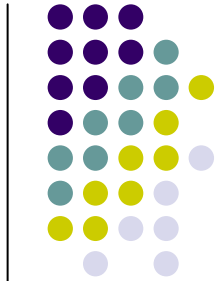
- Why do I do a presentation on iBATIS at a Oracle User Group? This sounds like something for Java or .NET developers than for Oracle DBA and Oracle Developer?

Introduction - cont



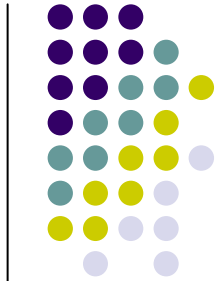
- Some of the struggles database developers faces.
  - Engaged late in the game. Database work is some times an after thought.
  - DB work is under estimated and usually sounded like this, “we just need some tables”, or “we just need some fields”.
  - Project is loosely spec’ed, poorly managed and deadline vastly underestimated.
  - And because of those, when the ball is in our hands the heat is usually on and there is not much time.

Introduction - cont



- This can cause from mild irritation to blame game and finger pointing when project is passed deadline and the heat is on.
- We are some times forced to make short sighted designs and cut corners which we will have to live with it for as long as you work there. Then in a few years, you will have a new comer asks, “why didn’t you …?”.

Introduction - cont

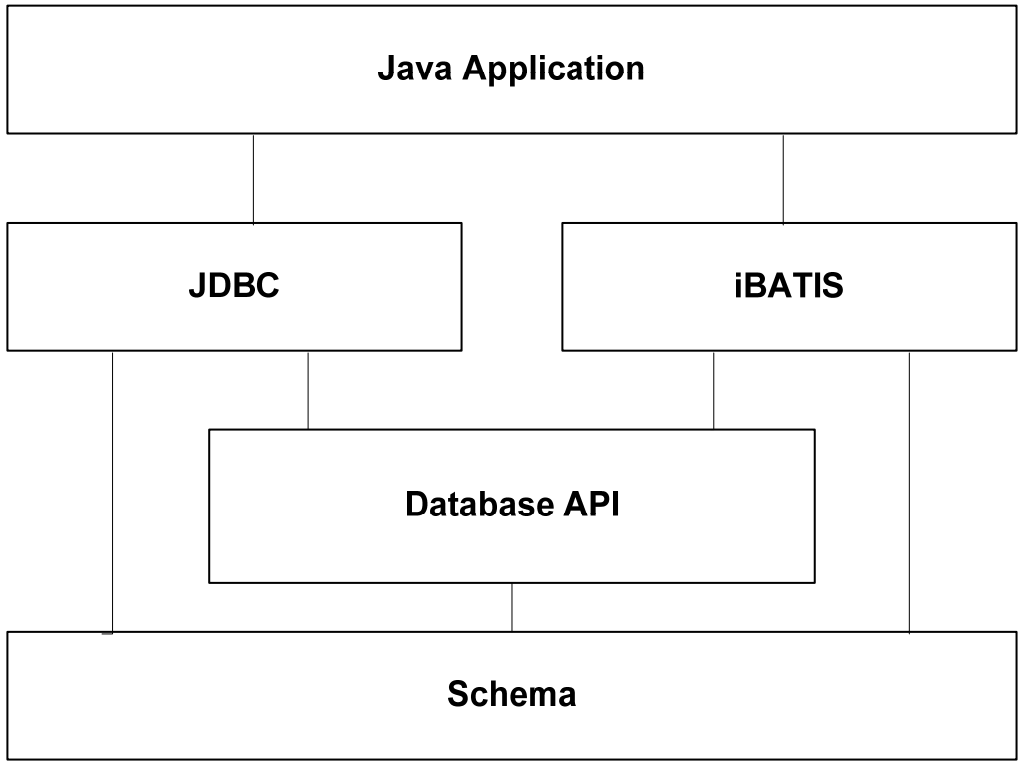


- What steps should we take?
  - Educate managers, project managers, and the application developers on the importance of database design.
  - Step up and speak out. Voice your concerns. Put it in writing and send to the key stack holders.
  - It’s time to change again.

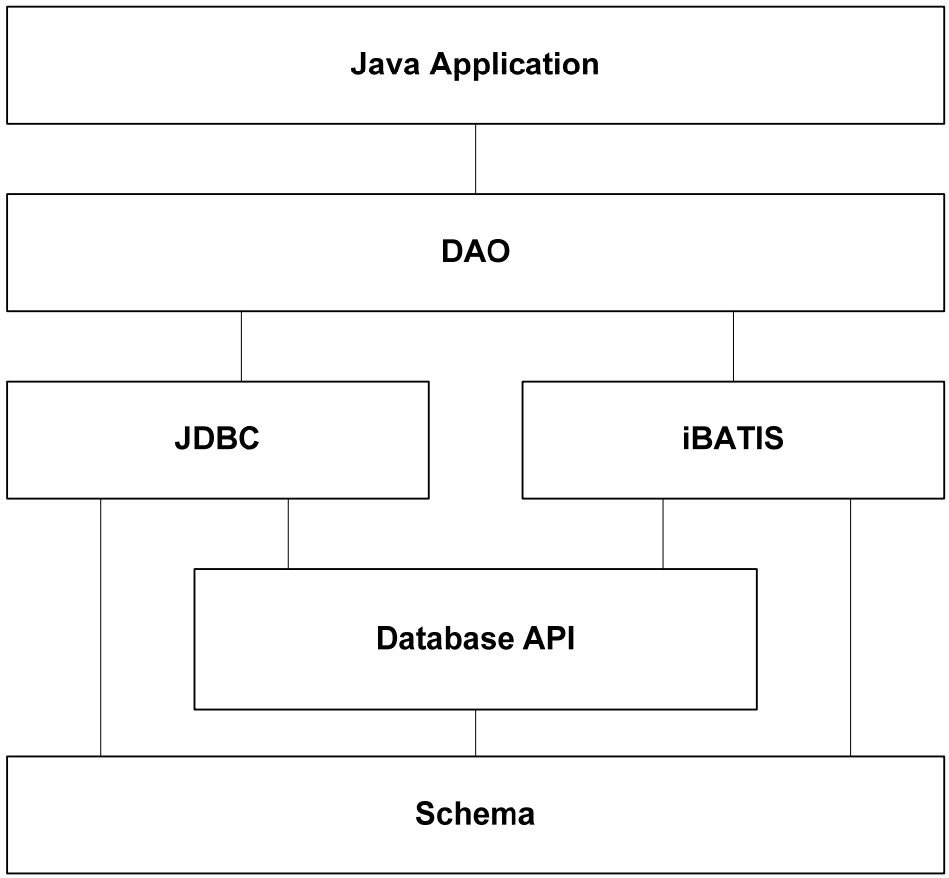
Introduction - cont

- Let’s look at some different methods of an Java application accesses Oracle.
  - Direct use of SQLJ and JDBC: embed SQL or call stored procedures.
  - By use DAO: which can use SQLJ, JDBC or iBATIS etc.

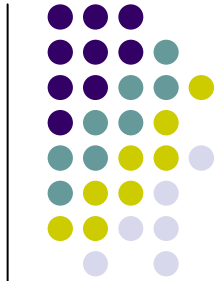
Introduction - cont



Introduction - cont



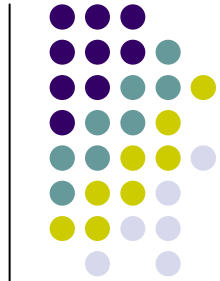
Introduction - cont



- Oracle Developer writing DAO?
  - DAO is the layer where application business logic and data access logic is separated.
- Previously Stored Procedures API is suppose to be the separation point.
  - It only works if the database developer has the control to dictate how application should access the data.
  - Application developer don’t know what to ask.

```
select deptno, cursor(select e.empno, e.ename from emp e
   where e.deptno = d.deptno) from dept d;
```

Introduction - cont

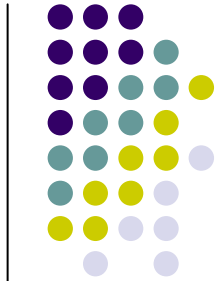


- OK, so maybe we don’t implement the DAO, but Oracle developers should at least review and approve the DAO.

Pros for iBATIS

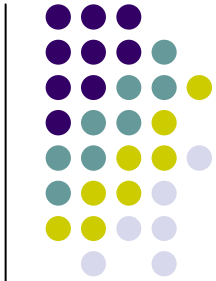
- All database access is done in the Data Mapper XML files. DBA, DB Developer can easily access SQL or Stored Procedure calls without coding a single line of JDBC.
- DBA and DB Developer can easily try out new SQL or Stored Procedures as long as it doesn’t modify the result.

Cons for iBATIS



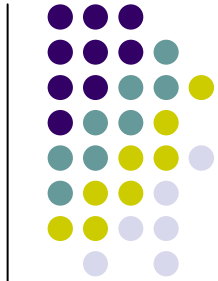
- Since iBATIS was mean to do 80% of the JDBC coding with only 20% of the code. There are certain things iBATIS doesn't support. Such as LOB access in which case you will have to code JDBC around it.

Setup



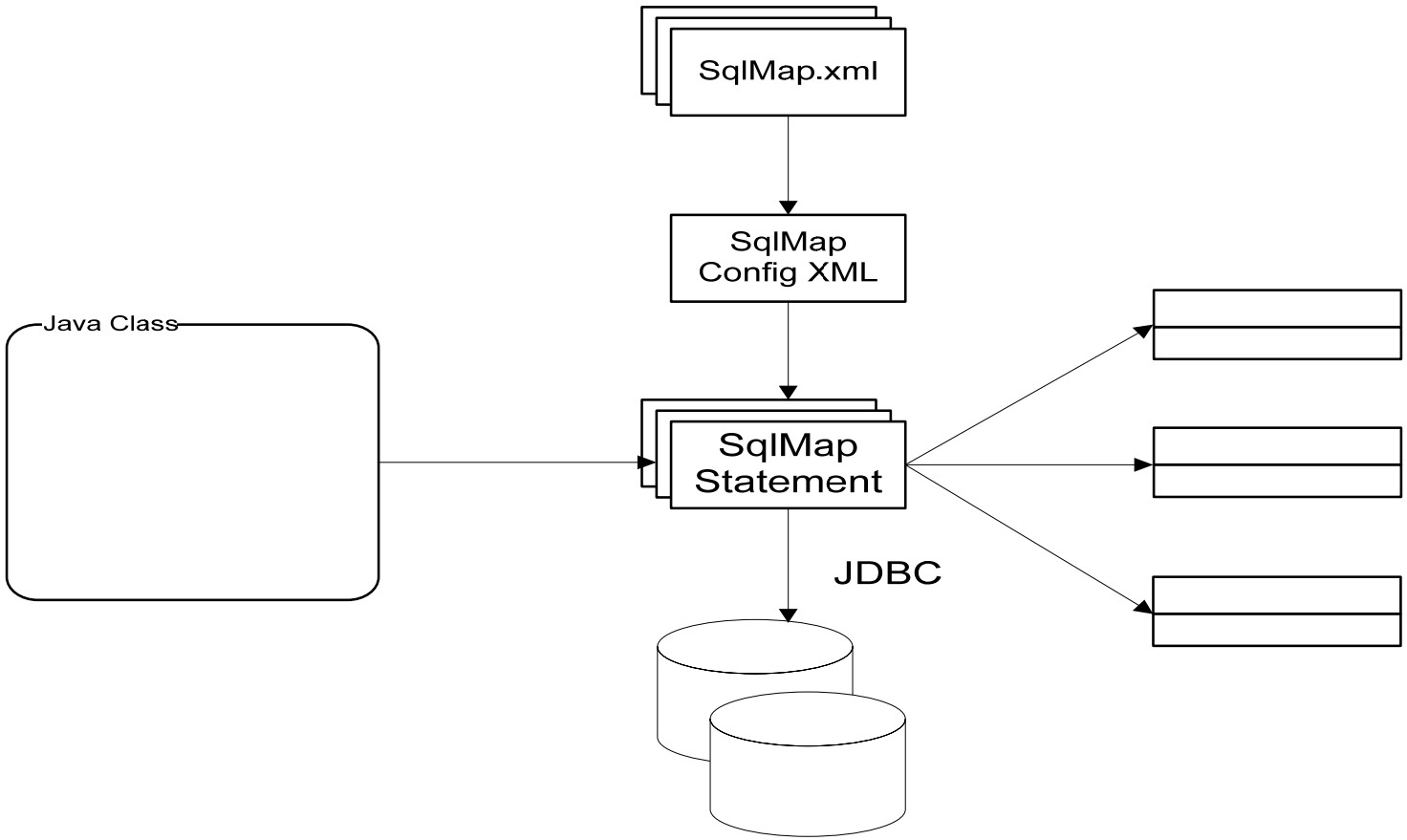
- What you need to run iBATIS:
  - JDK 1.5 or above. java.sun.com
  - Download iBATIS from ibatis.apache.org
  - Oracle JDBC driver, located at $ORACLE_HOME/jdbc/lib, or you can download from otn.oracle.com
    - 10g, ojdbc14.jar
    - 11g, ojdbc5.jar or ojdbc6.jar

Demo



- What do you need to run the demo?
  - Everything from the previous Setup slide.
  - Plus Oracle's scott/tiger schema. You can create it from $ORACLE_HOME/rdbms/admin/utlsampl.sql script.

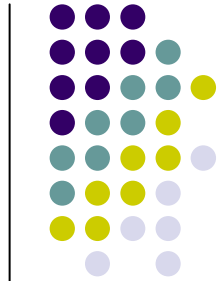
iBATIS Architecture



iBATIS Architecture - cont

- SqlMapConfig.xml
- SQL Map XML files. One XML file per object.
  - Employee.xml
  - Dept.xml

SqlMapConfig.xml



- SqlMapConfig.xml contains the DB connection information.
- It also contains a list of all the SQL Map XML files.

SqlConfigMap.xml - cont

```
<sqlMapConfig>
  <transactionManager type="JDBC"
   commitRequired="false">
    <dataSource type="SIMPLE">
      <property name="JDBC.Driver"
   value="oracle.jdbc.driver.OracleDriver"/>
      <property name="JDBC.ConnectionURL"
   value="jdbc:oracle:thin:@localhost:1521:ora11a"/>
      <property name="JDBC.Username" value="scott"/>
      <property name="JDBC.Password" value="tiger"/>
    </dataSource>
  </transactionManager>
  <sqlMap resource="Debug.xml"/>
  <sqlMap resource="Emp.xml"/>
</sqlMapConfig>
```

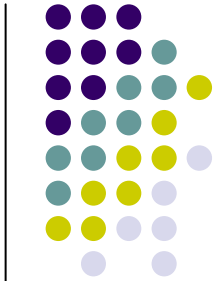
First iBATIS Program

- The best way to learn it is go through a sample application.
- First sample is without DAO and stored procedures to show iBATIS.

Java Bean

- Java Bean is an object that’s used to pass around data that’s a package of other objects and primitive types.

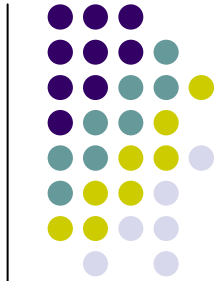
Employee Bean



```
public class Employee {
  private int empNo;
  private int manager;
  private int sal;
  private String empName;
  private String job;
  public Employee () {}
  public int getEmpNo() {
    return empNo;
  }
  public void setEmpNo(int empNo) {
    this.empNo = empNo;
  }
  public int getManager() {
    return manager;
  }
  public void setManager(int manager) {
    this.manager = manager;
  }
```

```
  public int getSal() {
     return sal;
   }
   public void setSal(int sal) {
     this.sal = sal;
   }
   public String getEmpName() {
     return empName;
   }
   public void setEmpName(String
     empName) {
     this.empName = empName;
   }
   public String getJob() {
     return job;
   }
   public void setJob(String job) {
     this.job = job;
   }
}
```

SQL Map



- Now that we have the Java Bean to hold employee information. It’s time to create the SQL to Java Object Map using the SQL Map file.

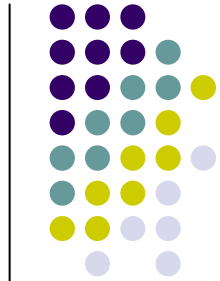
SQL Map Emp.xml : top

```
<?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="Emp">
… do select map
… etc
</sqlMap>
```

SQL Map Emp.xml : select

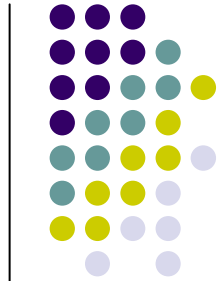
```
<resultMap id="EmpResult" class="Employee">
   <result property="empNo" column="EMPNO"/>
   <result property="manager" column="MGR"/>
   <result property="sal" column="SAL"/>
   <result property="empName" column="ENAME"/>
   <result property="job" column="JOB"/>
 </resultMap>
 <!-- Select with no parameters using the result
 map for Employee class. -->
 <select id="selectAllEmps"
 resultMap="EmpResult">
   select EMPNO, nvl(MGR,0) mgr, SAL, ENAME, JOB
 from emp
 </select>
```

SQL Map Emp.xml : select II



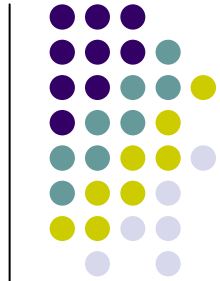
```
<!-- A simpler select example without the result
 map.  Note the aliases to match the properties
 of the target result class. -->
 <select id="selectEmpById" parameterClass="int"
 resultClass="Employee">
   select
     EMPNO as empNo,
     MGR as manager,
     SAL as sal,
     ENAME as empName,
     JOB as job
   from EMP
   where EMPNO = #id#
 </select>
```

SQL Map Emp.xml : insert



```
<!-- Insert example, using the Employee parameter
 class -->
 <insert id="insertEmp"
 parameterClass="Employee">
   insert into EMP (
     EMPNO, MGR, SAL, ENAME, JOB
   )
   values (
     #empNo#, #manager#, #sal#, #empName#, #job#
   )
 </insert>
```

SQL Map Emp.xml : update

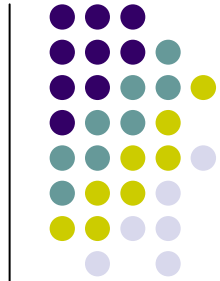


```
<!-- Update example, using the Employee parameter
 class -->
 <update id="updateEmp"
 parameterClass="Employee">
   update EMP set
     MGR = #manager#,
     SAL = #sal#,
     ENAME = #empName#,
     JOB = #job#
   where
     EMPNO = #empNo#
 </update>
```

SQL Map Emp.xml : delete

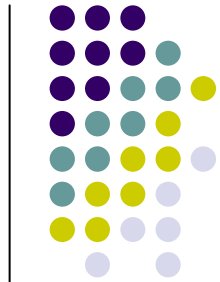
```
<!-- Delete example, using an integer as
 the parameter class -->
 <delete id="deleteEmpById"
 parameterClass="int">
   delete from EMP where EMPNO = #empNo#
 </delete>
```

Use our SQL Map



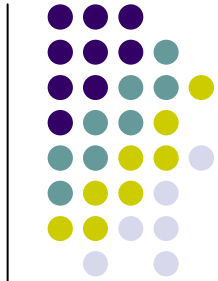
- Now that we have the Java Bean Employee and the necessary SQL Map. It’s time to write a small Java App to access DB through our first iBATIS SQL Map.

Java Test Program



```
private static SqlMapClient sqlMapper;
 static {
   try {
     Reader reader =
Resources.getResourceAsReader("SqlMapConfig.xml"
);
     sqlMapper =
SqlMapClientBuilder.buildSqlMapClient(reader);
     reader.close();
   } catch (IOException e) {
     throw new RuntimeException("Error building
the SqlMapClient instance." + e, e);
   }
 }
```

Java Test Program - cont



```
public static List selectAllEmps () throws SQLException {
  return sqlMapper.queryForList("selectAllEmps");
}
public static Employee selectEmpById  (int id) throws SQLException {
  return (Employee)sqlMapper.queryForObject("selectEmpById", id);
}
public static void insertEmp (Employee employee) throws SQLException
 {
  sqlMapper.insert("insertEmp", employee);
}
public static void updateEmp (Employee employee) throws SQLException
 {
  sqlMapper.update("updateEmp", employee);
}
public static void deleteEmpById (int id) throws SQLException {
  sqlMapper.delete("deleteEmpById", id);
}
```

Java Test Program - cont

```
 public static void main (String args[]) {
    Employee e1;
    try {
      System.out.println("Selecting from EMP:");
      Iterator it = selectAllEmps().iterator();
      while (it.hasNext()) {
        Employee emp = (Employee)it.next();
        System.out.println("empno=" + emp.getEmpNo() + "
   empName=" + emp.getEmpName() + " job=" + emp.getJob());
      }
}
    catch (SQLException se) {
      se.printStackTrace();
    }
  }
```

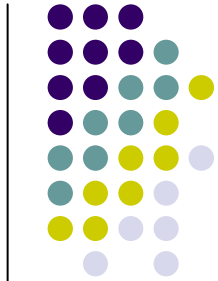
Java Test Program - cont

```
    System.out.println("Adding to EMP:");
    Employee emp = new Employee();
    emp.setEmpNo(1000);
emp.setManager(1000);      emp.setSal(100);
    emp.setEmpName("Test Emp");
    emp.setJob("CLERK");
    insertEmp(emp);
    System.out.println("EMP added.");
    e1 = selectEmpById(1000);
    System.out.println("empno=" + e1.getEmpNo()
+ " empName=" + e1.getEmpName() + " job=" +
e1.getJob());
    System.out.println("EMP added.");
```

Java Test Program - cont

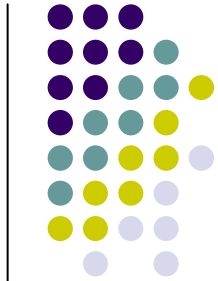
```
        System.out.println("Update EMP:");
        emp.setJob("MANAGER");
        updateEmp(emp);
        e1 = selectEmpById(1000);
        System.out.println("empno=" + e1.getEmpNo()
+ " empName=" + e1.getEmpName() + " job=" +
e1.getJob());
        System.out.println("EMP updated.");
        System.out.println("Delete EMP:");
        deleteEmpById(1000);
        System.out.println("EMP deleted.");
```

Run our first sample



- Compile the Employee.java java bean.
  - javac Employee.java
- Compile the test Java application.
  - javac TestEmp.java
- Run it.
  - java TestEmp

Recap

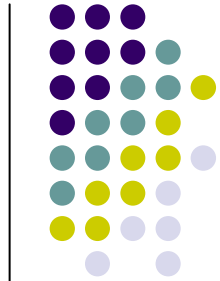


- What we did so far.
  - Created SqlMapConfig.xml
  - Created Java bean Employee.java
  - Created the Emp.xml SQL Map XML file.
  - Create the Java test application TestEmp.java to access the DB

DAO

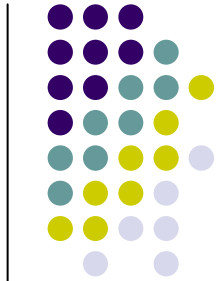
- We will reuse the Employee Java Bean we used.
- Create an DAO interface class.
- Create an DAO implementation class.

DAO Interface



- DAO Interface is like a contract between Java classes.
- DAO Interface is abstract.
- Interface only contains method signatures.
- Interface needs to be implemented.

DAO Interface Class



- Create EmpDAO.java

```
public interface EmpDAO {
    public abstract void add (final Employee
  employee) throws Exception;
    public abstract Employee get (final
  Integer uniqueID) throws Exception;
    public abstract void remove (final
  Integer uniqueID) throws Exception;
}
```

Oracle Stored Procedure

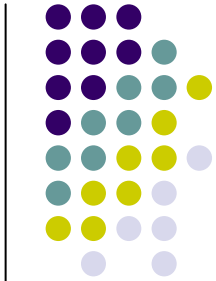
- Now before we implement our DAO Interface, we need the Oracle SP that will do the things for us.

Oralce SP : package

- Create a package called emp_util

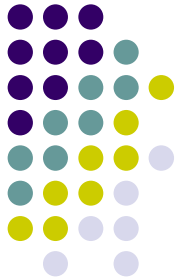
```
create or replace package body emp_util
as
```

Oracle SP : add



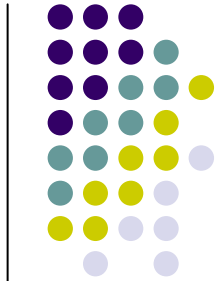
```
procedure add (p_empno in emp.empno%TYPE,
                 p_ename in emp.ename%TYPE,
                 p_job in emp.job%TYPE,
                 p_mgr in emp.mgr%TYPE,
                 p_sal in emp.sal%TYPE,
                 p_deptno in emp.deptno%TYPE,
                 status out number)
 is
   ckact                    number;
 begin
   insert into emp (empno, ename, job, mgr, hiredate, sal, deptno)
   values (p_empno, p_ename, p_job, p_mgr, sysdate, p_sal, p_deptno);
   status := SUCCESS;
 exception
   when others then
     dbms_output.put_line(SQLCODE || SQLERRM);
     status := FAILURE;
 end add;
```

Oracle SP : remove



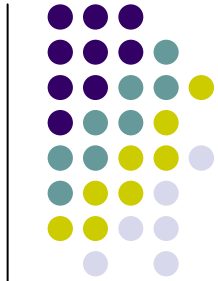
```
procedure remove (p_empno in emp.empno%TYPE,
                  status out number)
 is
   record_not_found    exception;
 begin
   delete from emp where empno = p_empno;
   if sql%rowcount = 0 then
     raise record_not_found;
   end if;
   status := SUCCESS;
 exception
   when record_not_found then
     status := FAILURE;
   when others then
     dbms_output.put_line(SQLCODE || SQLERRM);
     status := FAILURE;
 end remove;
```

Oracle SP : get



```
procedure get (status out number,
                  retrc out tRC)
 is
 begin
   open retrc for select EMPNO, nvl(MGR,0) mgr,
 SAL, ENAME, JOB, DEPTNO from emp;
   status := SUCCESS;
 exception
   when others then
     status := FAILURE;
 end get;
```

Oracle SP : get II



```
procedure get (p_empno in emp.empno%TYPE,
                  status out number,
                  retrc out tRC)
 is
 begin
   open retrc for select nvl(MGR,0) mgr, SAL,
 ENAME, JOB, DEPTNO from emp where empno =
 p_empno;
   status := SUCCESS;
 exception
   when others then
     status := FAILURE;
 end get;
```

SQL Map Employee XML

- Before we could start implementing the DAO, we also need the SQL Map XML file that will map Stored Procedures to Java Object.

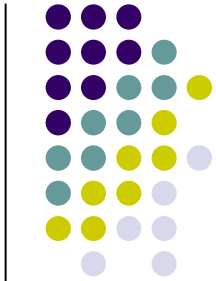
SQL Map Employee XML : add

```
<!-- ================================== -->
   <parameterMap id="addMap" class="java.util.Map">
       <parameter property="empNo" jdbcType="INTEGER"
  javaType="java.lang.Integer" mode="IN"/>
       <parameter property="empName" jdbcType="VARCHAR"
  javaType="java.lang.String" mode="IN"/>
       <parameter property="job" jdbcType="VARCHAR"
  javaType="java.lang.String" mode="IN"/>
       <parameter property="manager" jdbcType="INTEGER"
  javaType="java.lang.Integer" mode="IN"/>
       <parameter property="sal" jdbcType="INTEGER"
  javaType="java.lang.Integer" mode="IN"/>
       <parameter property="deptNo" jdbcType="INTEGER"
  javaType="java.lang.Integer" mode="IN"/>
       <parameter property="status" jdbcType="INTEGER"
  javaType="java.lang.Integer" mode="OUT"/>
    </parameterMap>
    <procedure id="add" parameterMap="addMap" >
       begin emp_util.add(?,?,?,?,?,?,?); end;
    </procedure>
```

SQL Map Employee XML : get

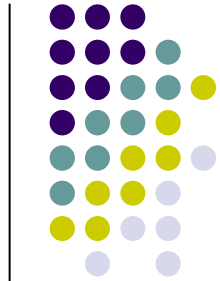
```
 <resultMap id="getEmpResultMap" class="Employee">
     <result property="manager" column="MGR" />
     <result property="sal" column="SAL" />
     <result property="empName" column="ENAME" />
     <result property="job" column="JOB" />
     <result property="deptNo" column="DEPTNO" />
 </resultMap>
 <parameterMap id="getEmpMap" class="java.util.HashMap" >
     <parameter property="empNo" jdbcType="INTEGER"
javaType="java.lang.Integer" mode="IN"/>
     <parameter property="status" jdbcType="INTEGER"
javaType="java.lang.Integer" mode="OUT"/>
     <parameter property="emp" jdbcType="ORACLECURSOR"
javaType="java.sql.ResultSet" mode="OUT"
resultMap="getEmpResultMap"/>
 </parameterMap>
 <procedure id="get" parameterMap="getEmpMap" >
     begin emp_util.get(?, ?, ?); end;
 </procedure>
```

SQL Map Employee XML : remove



```
<parameterMap id="removeMap"
 class="java.util.Map" >
      <parameter property="empNo"
 jdbcType="INTEGER" javaType="java.lang.Integer"
 mode="IN"/>
      <parameter property="status"
 jdbcType="INTEGER" javaType="java.lang.Integer"
 mode="OUT"/>
  </parameterMap>
  <procedure id="remove"
 parameterMap="removeMap" >
      begin emp_util.remove(?,?); end;
  </procedure>
```

EmpDAOImpl



- There can be as many DAO implementation as you want.
- We can have an implementation that use iBATIS.
- We can have another implementation that uses straight JDBC.
- We can have another implementation that uses Hibernate.

EmpDAOImpl - add

```
public void add (Employee employee) throws Exception {
     try {
       System.out.println("Going to invoke SqlMap: add");
       HashMap<String, Object> map = new HashMap<String, Object>(7);
       map.put("empNo", employee.getEmpNo());
       map.put("empName", employee.getEmpName());
       map.put("job", employee.getJob());
       map.put("manager", employee.getManager());
       map.put("sal", employee.getSal());
       map.put("deptNo", employee.getDeptNo());
       sqlMapper.update("add", map);
       System.out.println("After invoking SqlMap: add: " + map);
       Integer status = (Integer) map.get("status");
       if (status == null)
         throw new Exception("Recieved null status value");
       System.out.println("Added unique id: " + employee.getEmpNo());
     }
     catch (SQLException e) {
       throw new Exception("Executing get stored procedure", e);
     }
   }
```

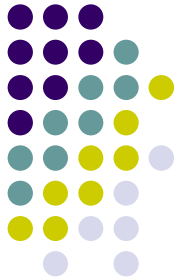
EmpDAOImpl - get

```
public Employee get (final Integer empNo) throws Exception {
     try {
       System.out.println("Going to invoke SqlMap: get");
       HashMap<String, Object> map = new HashMap<String, Object>();
       map.put("empNo", empNo);
       sqlMapper.update("get", map);
       System.out.println("After invoking SqlMap: get: " + map);
       Integer status = (Integer) map.get("status");
       if (status == null)
         throw new Exception("Recieved null status value");
       System.out.println("Got unique id: " + empNo);
       ArrayList<Employee> d = (ArrayList<Employee>) map.get("emp");
       if ((d == null) || d.size() > 1)
         throw new Exception("Recieved null user value");
       return d.get(0);
     } catch (SQLException e) {
       throw new Exception("Executing get stored procedure", e);
     }
   }
```

EmpDAOImpl - remove

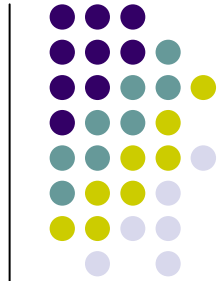
```
public void remove (final Integer empNo) throws Exception {
     try {
       HashMap<String, Object> map = new HashMap<String,
  Object>();
       map.put("empNo", empNo);
       sqlMapper.update("remove", map);
       System.out.println("After invoking SqlMap: remove: "
  + map);
       Integer status = (Integer) map.get("status");
       if (status == null)
         throw new Exception("Recieved null status value");
       System.out.println("Removed emp no : " + empNo);
     } catch (SQLException e) {
       throw new Exception("Executing remove stored
  procedure", e);
     }
   }
```

DAO Test Program



```
public final class EmpDAOTest {
  private static Employee emp, emp2;
  public static void main (String args[]) {
    EmpDAO d = new EmpDAOImpl();
    emp = new Employee();
    emp.setEmpNo(1000); emp.setManager(0); emp.setSal(1234);
    emp.setDeptNo(20); emp.setEmpName("Alvin");
    emp.setJob("MANAGER");
    System.out.println("test add :");
    try {d.add(emp);}
    catch (Exception e) { e.printStackTrace(); }
    System.out.println("test get :");
    try {emp2 = d.get(1000);}
    catch (Exception e) { e.printStackTrace(); }
    System.out.println(emp2.getEmpNo() + " " + emp2.getEmpName()
   + " " + emp2.getJob());
    System.out.println("test remove :");
    try {d.remove(1000);}
    catch (Exception e) { e.printStackTrace(); }
  }
}
```

Debug in iBATIS

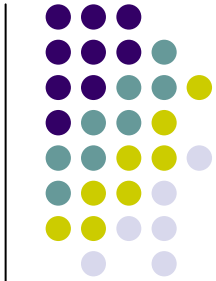


- iBATIS some times hides the exceptions and it is not obvious as to why things didn’t work as intended. This means you could miss seeing a SQLException that could of told you the exact ORA- error that happened.

Debug in iBATIS – debug.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMap PUBLIC "-//ibatis.apache.org//DTD SQL Map
   2.0//EN" "">
<sqlMap namespace="Debug">
    <!-- ======================================== -->
    <!-- Oracle debug statements                  -->
    <!-- ======================================== -->
    <update id="ev10046on">
        alter session set events '10046 trace name context
   forever, level 12'
    </update>
    <update id="ev10046off">
        alter session set events '10046 trace name context
   off'
    </update>
</sqlMap>
```

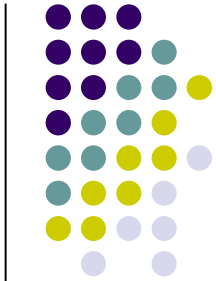
Debug in iBATIS – debug.xml



- To use it, in the DAO or where iBATIS calls are made just add the below two line around the code block you want to capture a trace.

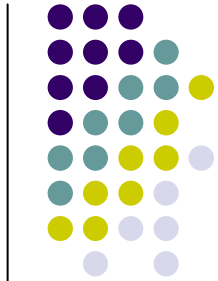
```
sqlMapper.update(“ev10046on”);
….
sqlMapper.update("ev10046off");
```

Using Data Source



- A note about using OracleDataSource
  - There is no way to set all Connections coming from a ODS auto commit off.
  - One must set auto commit off after get a Connection from a ODS.
  - Upon returning the Connection back to an ODS cache, future get Connection will return the Connection with auto commit on again.

Conclusion



- iBATIS is a wonderful tool, but just like all tools we use, it has its place. When used correctly it can save productivity and make things easier for both the application developers and Oracle developers.
- The most important thing is planning. If you can have the DAO interface well defined and clearly defined, then it’s easier for both the application developers and database developers.

Q&A