Tarena Technology
达内科技

Tarena Teaching System

MyBatis 核心

学员用书

Java企业应用及互联网
高级工程师培训课程

达内集团教学研发部 编著

目　录

达内IT培训集团

MyBatis 核心

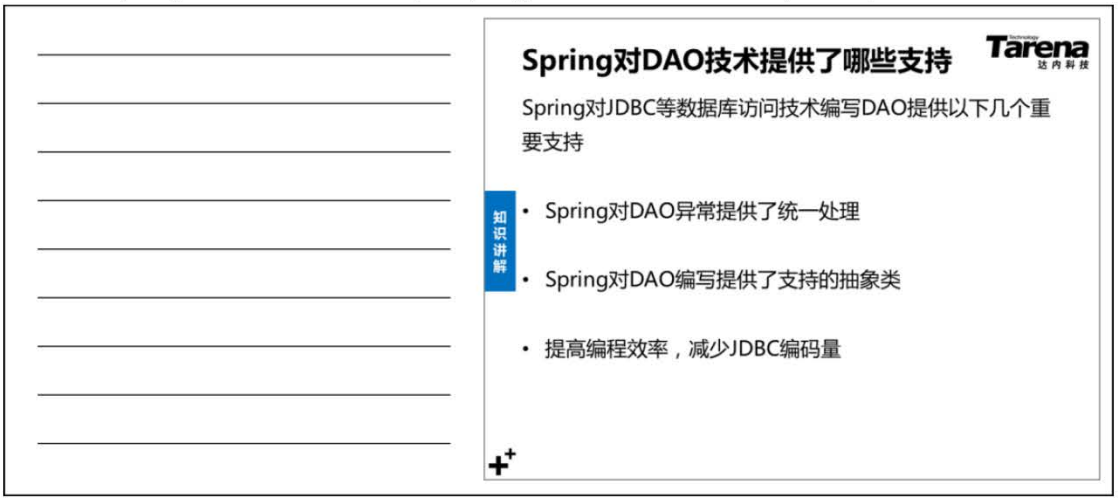
Unit01

达内IT培训集团

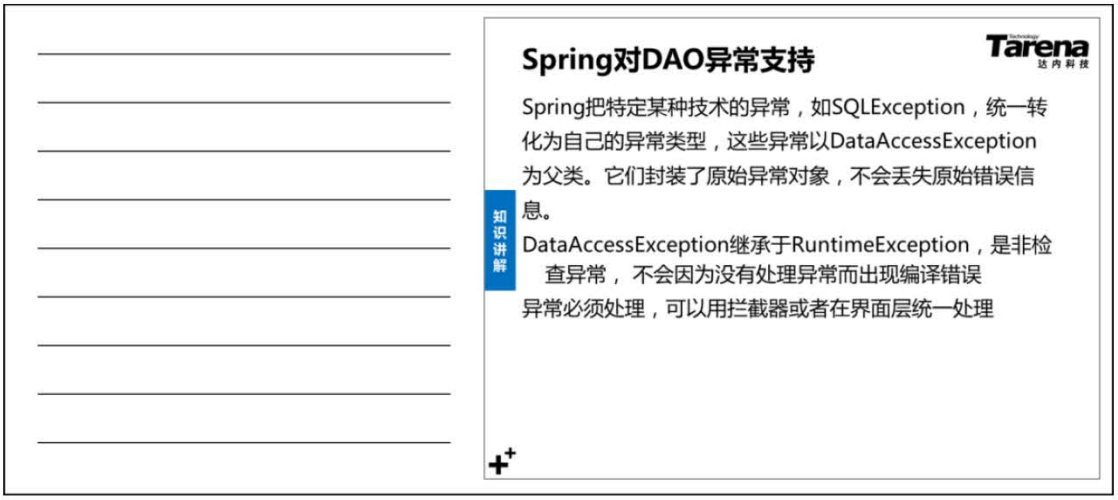
1. Spring + JDBC Template

1.1. Spring 对 JDBC 的整合支持

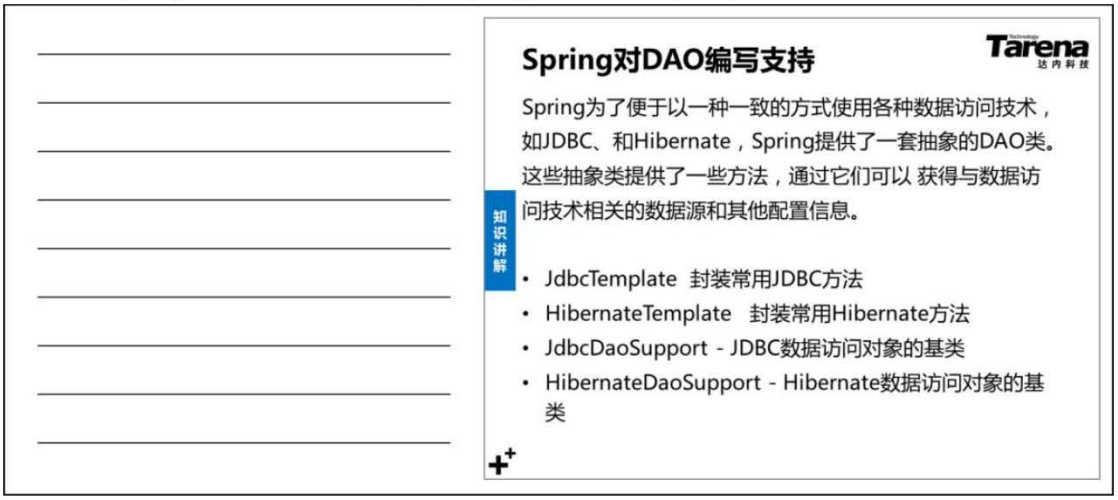
1.1.1. 【Spring 对 JDBC 的整合支持】Spring 对 DAO 技术提供了哪些支持



1.1.2. 【Spring 对 JDBC 的整合支持】Spring 对 DAO 异常支持

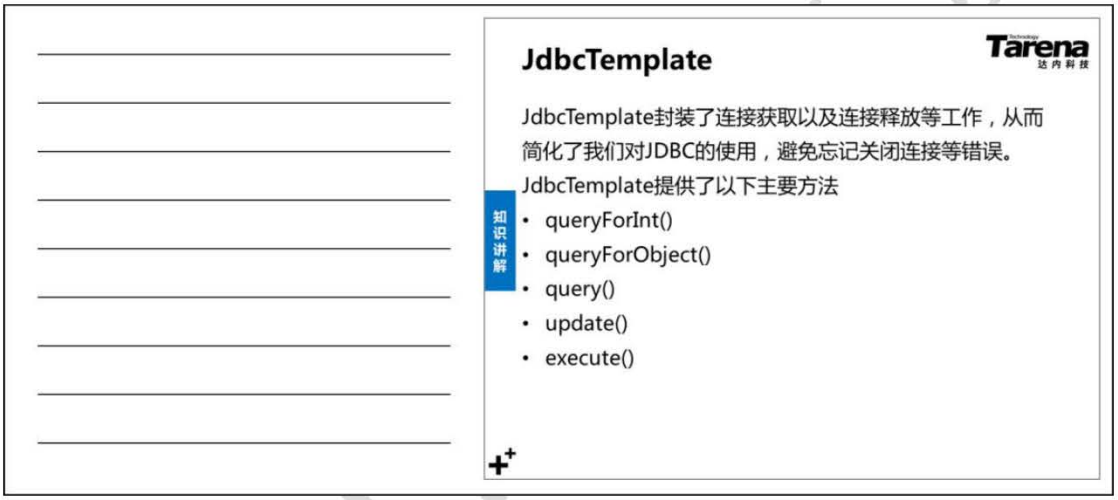


1.1.3. 【Spring 对 JDBC 的整合支持】Spring 对 DAO 编写支持



1.1.4. 【Spring 对 JDBC 的整合支持】JdbcDaoSupport

1.1.5. 【Spring 对 JDBC 的整合支持】JdbcTemplate



1.1.6. 【Spring 对 JDBC 的整合支持】如何编写 DAO 组件

如何编写DAO组件（续1）

不继承JdbcDaoSupport的用法—DAO实现类

解说知识

```
public class JdbcEmpDAO implements EmpDAO{
   private JdbcTemplate template;

   public void setTemplate(JdbcTemplate template){
       this.template = template;
   }

   public void add(Emp emp){
     String sql = "insert into ....";
     Object[] params = {emp.getName(),emp.getSalary...};
     template.update(sql,params);
   }
}
```

如何编写DAO组件（续2）

不继承JdbcDaoSupport的用法—DAO配置

解说知识

```
<bean id="myDataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
   <property name="driverClassName" value="..."/>
   <property name="url" value="..."/>
   <property name="username" value="..."/>
   <property name="password" value="..."/>
</bean>
<bean id="jdbcTemplate"
        class="org.springframework.jdbc.core.JdbcTemplate">
   <property name="dataSource" ref="myDataSource">
    </property>
</bean>
```

如何编写DAO组件（续3）

不继承JdbcDaoSupport的用法—DAO配置（续）

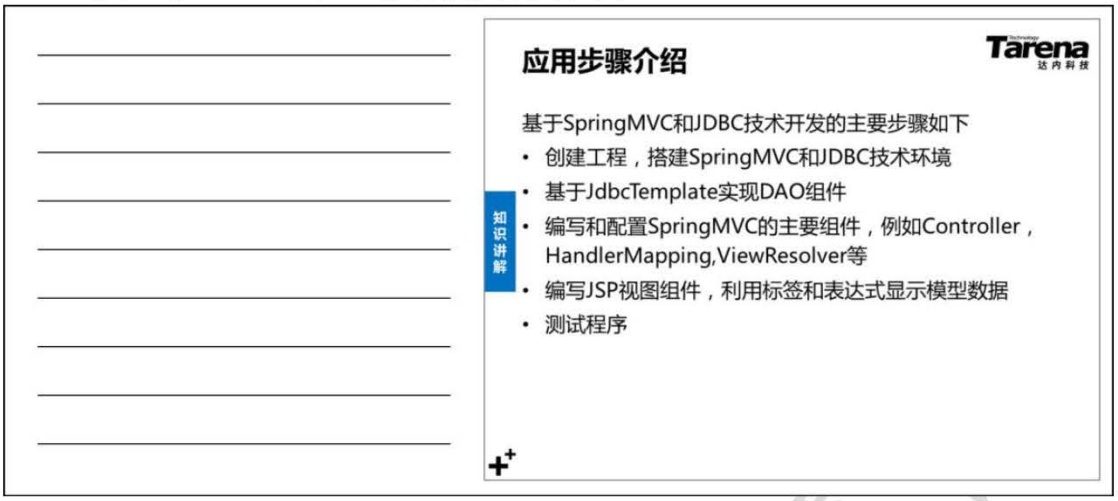
解说知识

```
<bean id="jdbcEmpDao" class="org.tarena.dao.JdbcEmpDAO">
   <property name="template" ref="jdbcTemplate">
   </property>
</bean>
```

也可以使用 @Resource 注解方式注入

1.2. Spring + JDBC Template 应用

1.2.1. 【Spring + JDBC Template 应用】应用步骤介绍



应用步骤介绍（续3）

Tarena 达内科技

如何编写和配置SpringMVC的主要组件？

- 编写Controller和请求处理方法
- 配置<mvc:annotation-driven/>，支持@RequestMapping
- 配置Controller组件
  - 开启组件扫描，将Controller扫描到Spring容器
  - 需要DAO时采用注入方式使用
  - 在请求处理方法上使用@RequestMapping指定对应的请求
- 配置ViewResolver组件

知识详解

应用步骤介绍（续4）

Tarena 达内科技

在JSP视图组件中如何显示模型数据？

- JSP可以使用JSTL标签库，需要引入开发包
- JSP可以使用EL表达式
- JSP可以使用SpringMVC的表单标签

知识详解

经典案例

1. 实现员工列表显示 DAO

- **问题**

利用 Spring 和 JDBC 技术整合实现对员工表 T_EMP 的增删改查操作的 DAO 组件。

- **方案**

实现 Spring 和 JDBC 技术整合有两种方法：

1、基于 JdbcDaoSupport 类编写 DAO，然后使用 JdbcTemplate 对象进行操作；

2、直接在 Spring 容器创建 JdbcTemplate，然后注入到 DAO 进行操作。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：在 MySQL 的 test 数据库中创建表和数据

请在 MySQL 的 test 数据库中执行以下 SQL 语句，用来创建表和相关数据：

```
   create table t_emp(
    empno int auto_increment primary key,
    ename varchar(20),
    job varchar(10),
    mgr int,
    hiredate date,
    sal double,
    comm double,
    deptno int
   );

   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(1,'SMITH','CLERK',3,'1980-5-12',800,null,20);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(2,'ALLEN','SALESMAN',3,'1981-6-3',1600,300,30);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(3,'WARD','SALESMAN',4,'1990-3-15',1250,500,30);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(4,'JONES','MANAGER',5,'1985-4-8',2975,null,20);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(5,'MARTIN','SALESMAN',7,'1986-3-8',1250,1400,30);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(6,'BLAKE','MANAGER',9,'1989-6-1',2850,null,30);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7,'CLARK','MANAGER',9,'1995-10-1',2450,null,10);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(8,'SCOTT','ANALYST',9,'1993-5-1',3000,null,20);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(9,'KING','PRESIDENT',null,'1988-8-8',5000,null,10);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(10,'TURNER','SALESMAN',5,'1983-2-1',1500,0,30);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(11,'ADAMS','CLERK',5,'1992-7-3',1100,null,20);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
```

```
(12,'JAMES','CLERK',1,'1996-9-10',950,null,30);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(13,'FORD','ANALYST',1,'1993-1-1',3000,null,20);
   Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(14,'MILLER','CLERK',3,'1983-10-9',1300,null,10);
```

步骤二：新建工程，导入 jar 包

新建名为 SpringJDBC_Day06_Part1 的 web 工程，在该工程导入如图-1 所示的 jar 包：

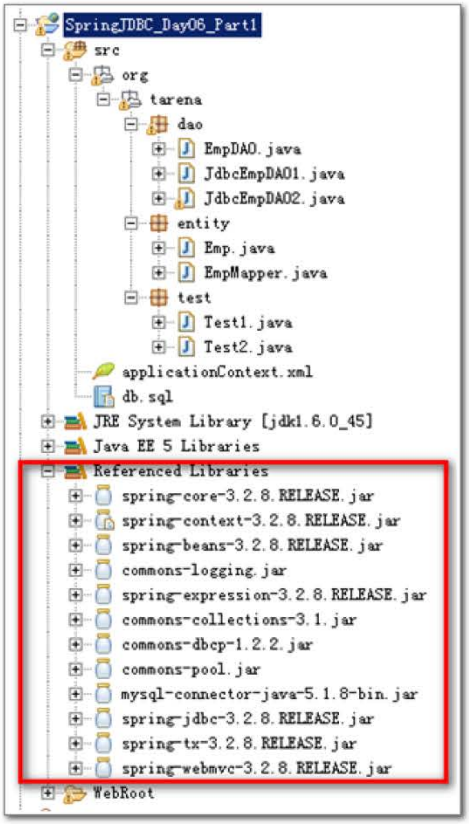


图-1

步骤三：新建 Spring 配置文件

新建 Spring 配置文件 applicationContext.xml。如图-2 所示：

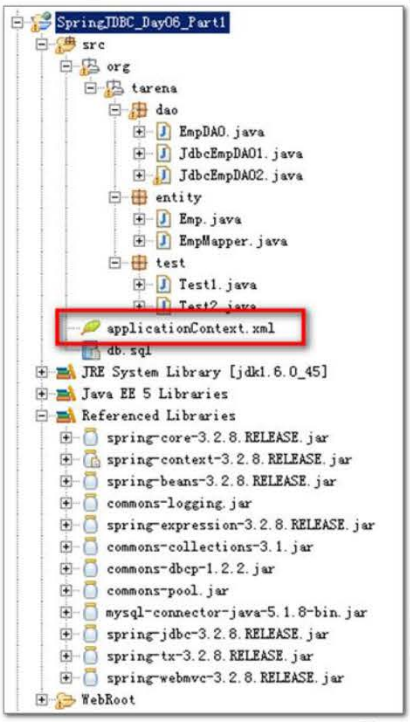


图-2

applicationContext.xml 文件中的代码如下所示：

```
    <?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:context="http://www.springframework.org/schema/context"
     xmlns:jdbc="http://www.springframework.org/schema/jdbc"
     xmlns:jee="http://www.springframework.org/schema/jee"
     xmlns:tx="http://www.springframework.org/schema/tx"
     xsi:schemaLocation="
         http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
         http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
         http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
         http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
         http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
         http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">


    <bean id="myDataSource"
         class="org.apache.commons.dbcp.BasicDataSource"
         destroy-method="close">
         <property name="driverClassName" value="com.mysql.jdbc.Driver" />
         <property name="url" value="jdbc:mysql://localhost:3306/test" />
         <property name="username" value="root" />
         <property name="password" value="root" />
    </bean>

    <!-- 继承 JdbcDaoSupport -->
    <bean id="jdbcEmpDao1" class="org.tarena.dao.JdbcEmpDAO1">
         <property name="dataSource" ref="myDataSource"></property>
```

```
    </bean>
</beans>
```

步骤四：新建 Emp 实体类和 EmpMapper 映射类

新建 Emp 实体类和 EmpMapper 映射类，如下图-3 所示：

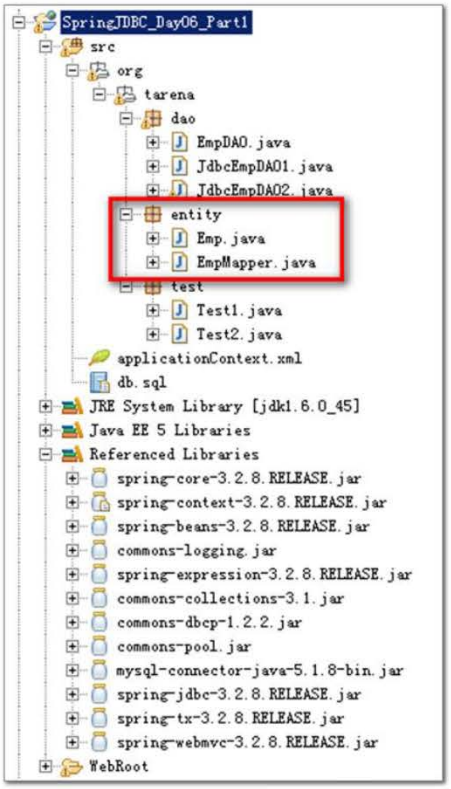


图-3

Emp 实体类代码如下：

```
package org.tarena.entity;

import java.sql.Date;

public class Emp {
 private Integer empno;
 private String ename;
 private String job;
 private Integer mgr;
 private Date hiredate;
 private Double sal;
 private Double comm;
 private Integer deptno;

 public Integer getEmpno() {
     return empno;
 }
 public void setEmpno(Integer empno) {
     this.empno = empno;
 }
 public String getEname() {
     return ename;
 }
 public void setEname(String ename) {
```

```
        this.ename = ename;
    }
    public String getJob() {
        return job;
    }
    public void setJob(String job) {
        this.job = job;
    }
    public Integer getMgr() {
        return mgr;
    }
    public void setMgr(Integer mgr) {
        this.mgr = mgr;
    }
    public Date getHiredate() {
        return hiredate;
    }
    public void setHiredate(Date hiredate) {
        this.hiredate = hiredate;
    }
    public Double getSal() {
        return sal;
    }
    public void setSal(Double sal) {
        this.sal = sal;
    }
    public Double getComm() {
        return comm;
    }
    public void setComm(Double comm) {
        this.comm = comm;
    }
    public Integer getDeptno() {
        return deptno;
    }
    public void setDeptno(Integer deptno) {
        this.deptno = deptno;
    }
}
```

EmpMapper 映射类代码如下：

```
package org.tarena.entity;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class EmpMapper implements RowMapper<Emp>{

    public Emp mapRow(ResultSet rs, int rowIndex) throws SQLException {
        Emp emp = new Emp();
        emp.setEmpno(rs.getInt("EMPNO"));
        emp.setEname(rs.getString("ENAME"));
        emp.setJob(rs.getString("JOB"));
        emp.setMgr(rs.getInt("MGR"));
        emp.setHiredate(rs.getDate("HIREDATE"));
        emp.setSal(rs.getDouble("SAL"));
        emp.setComm(rs.getDouble("COMM"));
        emp.setDeptno(rs.getInt("DEPTNO"));
        return emp;
    }
}
```

步骤五：新建 EmpDAO 接口

新建接口 EmpDAO，如下图-4 所示：

图-4

EmpDAO 接口的代码如下：

```
package org.tarena.dao;

import java.util.List;

import org.tarena.entity.Emp;

public interface EmpDAO {
 public void save(Emp emp);
 public void update(Emp emp);
 public void delete(int no);
 public Emp findByNo(int no);
 public List<Emp> findAll();
}
```

步骤六：新建 JdbcEmpDAO1 类

新建类 JdbcEmpDAO1，如图-5 所示：

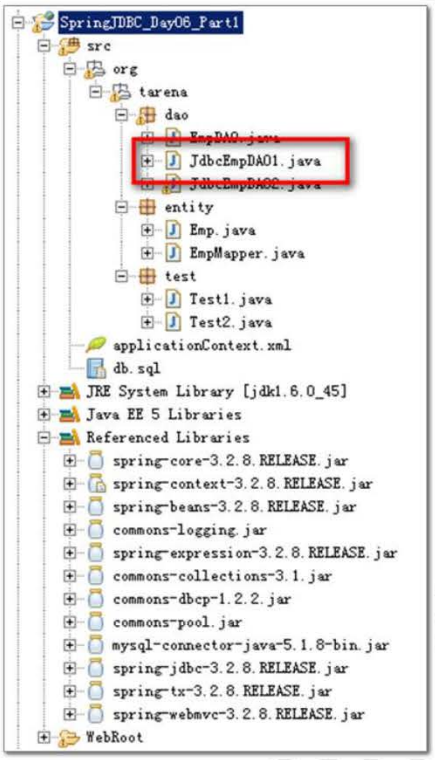


图-5

JdbcEmpDAO1 类的代码如下：

```
package org.tarena.dao;

import java.util.List;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import org.tarena.entity.Emp;
import org.tarena.entity.EmpMapper;

public class JdbcEmpDAO1 extends JdbcDaoSupport implements EmpDAO{
 public void delete(int no) {
     String sql = "delete from t_emp where empno=?";
     Object[] params = {no};
     super.getJdbcTemplate().update(sql,params);
 }

 public List<Emp> findAll() {
     String sql = "select * from t_emp";
     RowMapper<Emp> mapper = new EmpMapper();
     List<Emp> list = super.getJdbcTemplate().query(sql, mapper);
     return list;
 }

 public Emp findByNo(int no) {
     String sql = "select * from t_emp where EMPNO=?";
     Object[] params = {no};
     RowMapper<Emp> mapper = new EmpMapper();
     Emp emp = super.getJdbcTemplate().queryForObject(sql, params,mapper);
     return emp;
 }

 public void save(Emp emp) {
     String sql =
"insert  into  t_emp  (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)  "  +
```

```
    "values (?,?,?,?,?,?,?,?)";
        Object[] params = {
            emp.getEmpno(),
            emp.getEname(),
            emp.getJob(),
            emp.getMgr(),
            emp.getHiredate(),
            emp.getSal(),
            emp.getComm(),
            emp.getDeptno()
        };
        super.getJdbcTemplate().update(sql,params);
    }

    public void update(Emp emp) {
        String sql = "update t_emp " +
                "set ENAME=?,JOB=?,MGR=?," +
                "HIREDATE=?,SAL=?,COMM=?,DEPTNO=? " +
                "where EMPNO=?";
        Object[] params = {
                emp.getEname(),
                emp.getJob(),
                emp.getMgr(),
                emp.getHiredate(),
                emp.getSal(),
                emp.getComm(),
                emp.getDeptno(),
                emp.getEmpno()
        };
        super.getJdbcTemplate().update(sql,params);
    }
}
```

步骤七：新建 Test1 类

新建 Test1 类，如下图-6 所示：

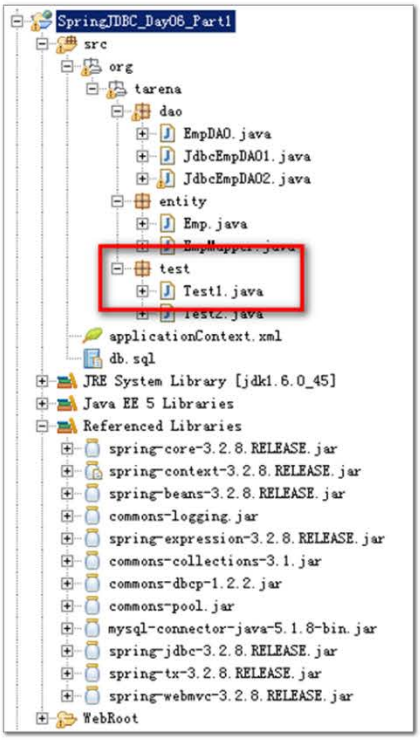


图-6

Test1 类代码如下：

```
package org.tarena.test;

import java.util.List;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

public class Test1 {

 /**
  * @param args
  */
 public static void main(String[] args) throws Exception{
     String conf = "applicationContext.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     EmpDAO empDao = ac.getBean("jdbcEmpDao1",EmpDAO.class);
     List<Emp> list = empDao.findAll();
     for(Emp emp : list){
         System.out.println(emp.getEmpno() +" "+emp.getEname());
     }
 }
}
```

步骤八：运行 Test1 类

运行 Test1 类，控制台输出结果如下图-7 所示：

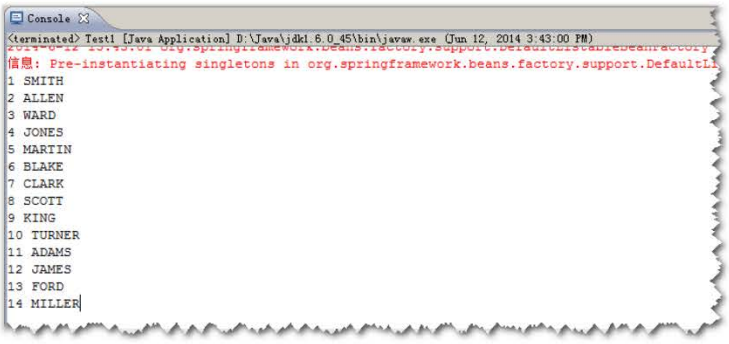


图-7

控制台输出如上图所示的信息，说明继承 JdbcDaoSupport 的 DAO 测试成功。

步骤九：创建 JdbcEmpDAO2 类

创建类 JdbcEmpDAO2，如图-8 所示：

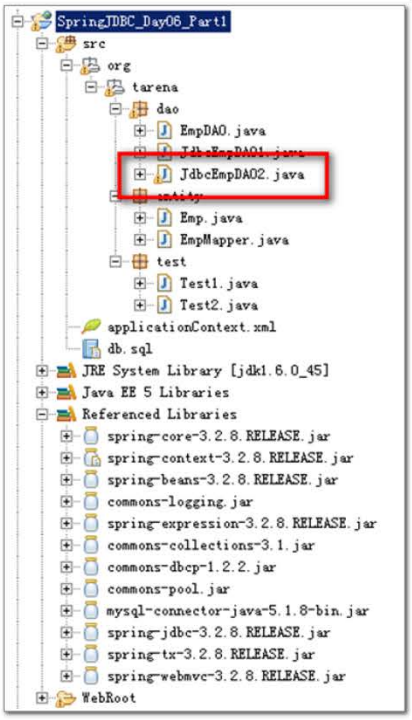


图-8

JdbcEmpDAO2 类代码如下：

```
package org.tarena.dao;

import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import org.tarena.entity.Emp;
import org.tarena.entity.EmpMapper;

public class JdbcEmpDAO2 implements EmpDAO{

 private JdbcTemplate template;
 public void setTemplate(JdbcTemplate template){
     this.template = template;
 }

 public void delete(int no) {
     String sql = "delete from t_emp where empno=?";
     Object[] params = {no};
     template.update(sql,params);
 }

 public List<Emp> findAll() {
     String sql = "select * from t_emp";
     RowMapper<Emp> mapper = new EmpMapper();
     List<Emp> list = template.query(sql, mapper);
     return list;
 }

 public Emp findByNo(int no) {
     String sql = "select * from t_emp where EMPNO=?";
     Object[] params = {no};
     RowMapper<Emp> mapper = new EmpMapper();
     Emp emp = template.queryForObject(sql, params,mapper);
```

```
        return emp;
    }

    public void save(Emp emp) {
        String sql =
    "insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) " +
                "values (?,?,?,?,?,?,?,?)";
        Object[] params = {
            emp.getEmpno(),
            emp.getEname(),
            emp.getJob(),
            emp.getMgr(),
            emp.getHiredate(),
            emp.getSal(),
            emp.getComm(),
            emp.getDeptno()
        };
        template.update(sql,params);
    }

    public void update(Emp emp) {
        String sql = "update t_emp " +
                "set ENAME=?,JOB=?,MGR=?," +
                "HIREDATE=?,SAL=?,COMM=?,DEPTNO=? " +
                "where EMPNO=?";
        Object[] params = {
                emp.getEname(),
                emp.getJob(),
                emp.getMgr(),
                emp.getHiredate(),
                emp.getSal(),
                emp.getComm(),
                emp.getDeptno(),
                emp.getEmpno()
        };
        template.update(sql,params);
    }
    }
```

步骤十：创建 Test2 类

创建类 Test2，如图-9 所示：

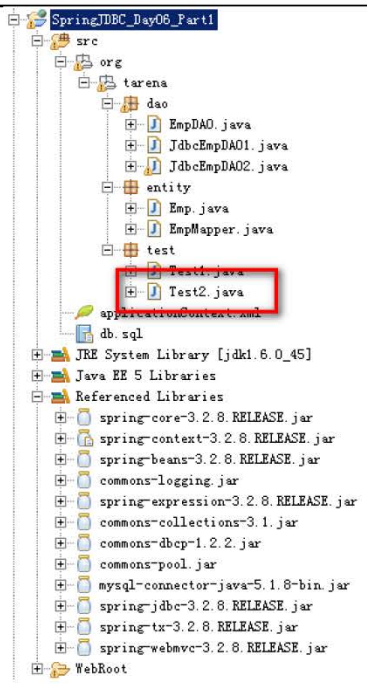


图-9

Test2 类代码如下：

```
package org.tarena.test;

import java.util.List;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

public class Test2 {

 /**
  * @param args
  */
 public static void main(String[] args) throws Exception{
     String conf = "applicationContext.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     EmpDAO empDao = ac.getBean("jdbcEmpDao2",EmpDAO.class);
     List<Emp> list = empDao.findAll();
     for(Emp emp : list){
         System.out.println(emp.getEmpno() +" "+emp.getEname());
     }
 }
}
```

步骤十一：修改 applicationContext.xml 文件

修改 applicationContext.xml，加入方框部分 XML 配置信息，如图-10 所示：

```
    <!-- 继承JdbcDaoSupport -->
    <bean id="jdbcEmpDao1" class="org.tarena.dao.JdbcEmpDAO1">
        <property name="dataSource" ref="myDataSource"></property>
    </bean>


    <!-- 不继承JdbcDaoSupport -->
    <bean id="jdbcTemplate"
        class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="myDataSource"></property>
    </bean>

    <bean id="jdbcEmpDao2" class="org.tarena.dao.JdbcEmpDAO2">
        <property name="template" ref="jdbcTemplate"></property>
    </bean>

</beans>
```

图-10

XML 配置如下：

```
<!-- 不继承 JdbcDaoSupport -->
<bean id="jdbcTemplate"
    class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="myDataSource"></property>
</bean>

<bean id="jdbcEmpDao2" class="org.tarena.dao.JdbcEmpDAO2">
    <property name="template" ref="jdbcTemplate"></property>
</bean>
```

步骤十二：运行 Test2 类

运行 Test2 类，控制台输入结果如下图-11 所示：

```
Console
<terminated> Test2 [Java Application] D:\Java\jdk1.6.0_45\bin\javaw.exe (Jun 12, 2014 4:00:49 PM)
2014-6-12 16:00:51 org.springframework.context.support.AbstractApplicationContext prepareRefresh
信息: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@bf32c: startup date [Thu Jun 12 16
2014-6-12 16:00:52 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
信息: Loading XML bean definitions from class path resource [applicationContext.xml]
2014-6-12 16:00:52 org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
信息: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@80fa6f: defi
1 SMITH
2 ALLEN
3 WARD
4 JONES
5 MARTIN
6 BLAKE
7 CLARK
8 SCOTT
9 KING
10 TURNER
11 ADAMS
12 JAMES
13 FORD
14 MILLER
```

图-11

控制台输出如上图所示的信息，说明不继承 JdbcDaoSupport 的 DAO 测试成功。

- **完整代码**

我们使用以下数据库结构：

```
create table t_emp(
 empno int auto increment primary key,
 ename varchar(20),
 job varchar(10),
 mgr int,
 hiredate date,
 sal double,
 comm double,
 deptno int
);

Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(1,'SMITH','CLERK',3,'1980-5-12',800,null,20);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(2,'ALLEN','SALESMAN',3,'1981-6-3',1600,300,30);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(3,'WARD','SALESMAN',4,'1990-3-15',1250,500,30);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(4,'JONES','MANAGER',5,'1985-4-8',2975,null,20);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(5,'MARTIN','SALESMAN',7,'1986-3-8',1250,1400,30);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(6,'BLAKE','MANAGER',9,'1989-6-1',2850,null,30);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7,'CLARK','MANAGER',9,'1995-10-1',2450,null,10);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(8,'SCOTT','ANALYST',9,'1993-5-1',3000,null,20);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(9,'KING','PRESIDENT',null,'1988-8-8',5000,null,10);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(10,'TURNER','SALESMAN',5,'1983-2-1',1500,0,30);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(11,'ADAMS','CLERK',5,'1992-7-3',1100,null,20);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(12,'JAMES','CLERK',1,'1996-9-10',950,null,30);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(13,'FORD','ANALYST',1,'1993-1-1',3000,null,20);
Insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(14,'MILLER','CLERK',3,'1983-10-9',1300,null,10);
```

applicationContext.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:jdbc="http://www.springframework.org/schema/jdbc"
 xmlns:jee="http://www.springframework.org/schema/jee"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xsi:schemaLocation="
     http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
     http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
     http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
     http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
     http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
```

```
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">

    <bean id="myDataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="root" />
    </bean>

    <!-- 继承 JdbcDaoSupport -->
    <bean id="jdbcEmpDao1" class="org.tarena.dao.JdbcEmpDAO1">
        <property name="dataSource" ref="myDataSource"></property>
    </bean>

    <!-- 不继承 JdbcDaoSupport -->
    <bean id="jdbcTemplate"
        class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="myDataSource"></property>
    </bean>

    <bean id="jdbcEmpDao2" class="org.tarena.dao.JdbcEmpDAO2">
        <property name="template" ref="jdbcTemplate"></property>
    </bean>

   </beans>
```

接口 org.tarena.dao.EmpDAO 代码如下：

```
   package org.tarena.dao;

   import java.util.List;
   import org.tarena.entity.Emp;

   public interface EmpDAO {
    public void save(Emp emp);
    public void update(Emp emp);
    public void delete(int no);
    public Emp findByNo(int no);
    public List<Emp> findAll();
    }
```

类 org.tarena.dao.JdbcEmpDAO1 代码如下：

```
   package org.tarena.dao;

   import java.util.List;
   import org.springframework.jdbc.core.RowMapper;
   import org.springframework.jdbc.core.support.JdbcDaoSupport;
   import org.tarena.entity.Emp;
   import org.tarena.entity.EmpMapper;

   public class JdbcEmpDAO1 extends JdbcDaoSupport implements EmpDAO{

    public void delete(int no) {
        String sql = "delete from t_emp where empno=?";
        Object[] params = {no};
        super.getJdbcTemplate().update(sql,params);
    }

    public List<Emp> findAll() {
        String sql = "select * from t_emp";
```

```
        RowMapper<Emp> mapper = new EmpMapper();
        List<Emp> list = super.getJdbcTemplate().query(sql, mapper);
        return list;
    }

    public Emp findByNo(int no) {
        String sql = "select * from t emp where EMPNO=?";
        Object[] params = {no};
        RowMapper<Emp> mapper = new EmpMapper();
        Emp emp = super.getJdbcTemplate().queryForObject(sql, params,mapper);
        return emp;
    }

    public void save(Emp emp) {
        String sql =
    "insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) " +
                "values (?,?,?,?,?,?,?,?)";
        Object[] params = {
            emp.getEmpno(),
            emp.getEname(),
            emp.getJob(),
            emp.getMgr(),
            emp.getHiredate(),
            emp.getSal(),
            emp.getComm(),
            emp.getDeptno()
        };
        super.getJdbcTemplate().update(sql,params);
    }

    public void update(Emp emp) {
        String sql = "update t emp " +
                "set ENAME=?,JOB=?,MGR=?," +
                "HIREDATE=?,SAL=?,COMM=?,DEPTNO=? " +
                "where EMPNO=?";
        Object[] params = {
                emp.getEname(),
                emp.getJob(),
                emp.getMgr(),
                emp.getHiredate(),
                emp.getSal(),
                emp.getComm(),
                emp.getDeptno(),
                emp.getEmpno()
        };
        super.getJdbcTemplate().update(sql,params);
    }
    }
```

类 org.tarena.dao.JdbcEmpDAO2 代码如下：

```
    package org.tarena.dao;

    import java.util.List;
    import org.springframework.jdbc.core.JdbcTemplate;
    import org.springframework.jdbc.core.RowMapper;
    import org.springframework.jdbc.core.support.JdbcDaoSupport;
    import org.tarena.entity.Emp;
    import org.tarena.entity.EmpMapper;

    public class JdbcEmpDAO2 implements EmpDAO{

     private JdbcTemplate template;
     public void setTemplate(JdbcTemplate template){
         this.template = template;
     }
```

```
    public void delete(int no) {
        String sql = "delete from t emp where empno=?";
        Object[] params = {no};
        template.update(sql,params);
    }

    public List<Emp> findAll() {
        String sql = "select * from t emp";
        RowMapper<Emp> mapper = new EmpMapper();
        List<Emp> list = template.query(sql, mapper);
        return list;
    }

    public Emp findByNo(int no) {
        String sql = "select * from t emp where EMPNO=?";
        Object[] params = {no};
        RowMapper<Emp> mapper = new EmpMapper();
        Emp emp = template.queryForObject(sql, params,mapper);
        return emp;
    }

    public void save(Emp emp) {
        String sql =
    "insert into t_emp (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) " +
                "values (?,?,?,?,?,?,?,?)";
        Object[] params = {
            emp.getEmpno(),
            emp.getEname(),
            emp.getJob(),
            emp.getMgr(),
            emp.getHiredate(),
            emp.getSal(),
            emp.getComm(),
            emp.getDeptno()
        };
        template.update(sql,params);
    }

    public void update(Emp emp) {
        String sql = "update t emp " +
                "set ENAME=?,JOB=?,MGR=?," +
                "HIREDATE=?,SAL=?,COMM=?,DEPTNO=? " +
                "where EMPNO=?";
        Object[] params = {
                emp.getEname(),
                emp.getJob(),
                emp.getMgr(),
                emp.getHiredate(),
                emp.getSal(),
                emp.getComm(),
                emp.getDeptno(),
                emp.getEmpno()
        };
        template.update(sql,params);
    }
    }
```

类 org.tarena.entity.Emp 代码如下：

```
    package org.tarena.entity;
    import java.sql.Date;

    public class Emp {
     private Integer empno;
     private String ename;
     private String job;
```

```
    private Integer mgr;
    private Date hiredate;
    private Double sal;
    private Double comm;
    private Integer deptno;

    public Integer getEmpno() {
        return empno;
    }
    public void setEmpno(Integer empno) {
        this.empno = empno;
    }
    public String getEname() {
        return ename;
    }
    public void setEname(String ename) {
        this.ename = ename;
    }
    public String getJob() {
        return job;
    }
    public void setJob(String job) {
        this.job = job;
    }
    public Integer getMgr() {
        return mgr;
    }
    public void setMgr(Integer mgr) {
        this.mgr = mgr;
    }
    public Date getHiredate() {
        return hiredate;
    }
    public void setHiredate(Date hiredate) {
        this.hiredate = hiredate;
    }
    public Double getSal() {
        return sal;
    }
    public void setSal(Double sal) {
        this.sal = sal;
    }
    public Double getComm() {
        return comm;
    }
    public void setComm(Double comm) {
        this.comm = comm;
    }
    public Integer getDeptno() {
        return deptno;
    }
    public void setDeptno(Integer deptno) {
        this.deptno = deptno;
    }
}
```

类 org.tarena.entity.EmpMapper 代码如下：

```
package org.tarena.entity;
import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class EmpMapper implements RowMapper<Emp>{
```

```
    public Emp mapRow(ResultSet rs, int arg1) throws SQLException {
        Emp emp = new Emp();
        emp.setEmpno(rs.getInt("EMPNO"));
        emp.setEname(rs.getString("ENAME"));
        emp.setJob(rs.getString("JOB"));
        emp.setMgr(rs.getInt("MGR"));
        emp.setHiredate(rs.getDate("HIREDATE"));
        emp.setSal(rs.getDouble("SAL"));
        emp.setComm(rs.getDouble("COMM"));
        emp.setDeptno(rs.getInt("DEPTNO"));
        return emp;
    }
}
```

类 org.tarena.test.Test1 代码如下：

```
package org.tarena.test;

import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

public class Test1 {
    /**
     * @param args
     */
    public static void main(String[] args) throws Exception{
        String conf = "applicationContext.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        EmpDAO empDao = ac.getBean("jdbcEmpDao1",EmpDAO.class);
        List<Emp> list = empDao.findAll();
        for(Emp emp : list){
            System.out.println(emp.getEmpno() +" "+emp.getEname());
        }
    }
}
```

类 org.tarena.test.Test2 代码如下：

```
package org.tarena.test;

import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

public class Test2 {
    /**
     * @param args
     */
    public static void main(String[] args) throws Exception{
        String conf = "applicationContext.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        EmpDAO empDao = ac.getBean("jdbcEmpDao2",EmpDAO.class);
        List<Emp> list = empDao.findAll();
        for(Emp emp : list){
            System.out.println(emp.getEmpno() +" "+emp.getEname());
        }
    }
}
```

2. Spring+JDBCTemplate 实现员工列表显示

- **问题**

基于 Spring 和 JDBCTemplate 技术实现员工列表功能，基于上一个案例知识点追加 Spring Web MVC 应用。

- **方案**

实现员工列表页面功能，关键过程如下：

1、基于 Spring 和 JdbcTemplate 整合，实现对员工表的查询操作；

2、追加 Spring Web MVC 功能实现请求到响应的处理；

3、基于 Spring IoC 完成 DAO 组件和 Controller 组件的注入。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：在 MySQL 的 test 数据库中创建表和数据

请确认在 MySQL 的 test 数据库中存在 SpringJDBC_Day06_Part1 练习中创建的表和数据，如果没有相关表和数据请执行 SpringJDBC_Day06_Part1 练习中相关 SQL 语句。

步骤二：新建工程，导入 jar 包

新建名为 SpringJDBC_Day06_Part2 的 web 工程,在该工程导入如图-12 所示的 jar 包：

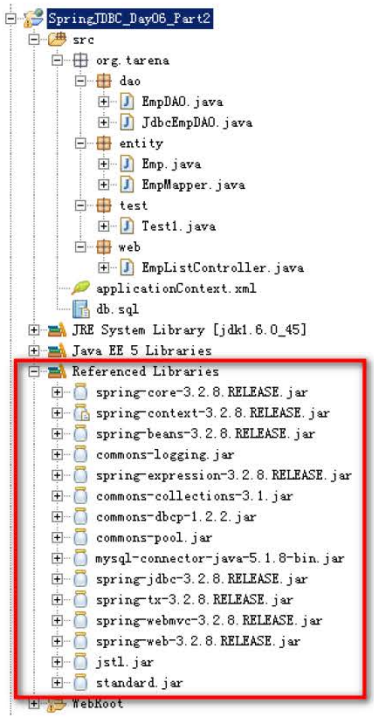


图-12

步骤三：新建 Spring 配置文件

新建 Spring 配置文件 applicationContext.xml。如图-13 所示：

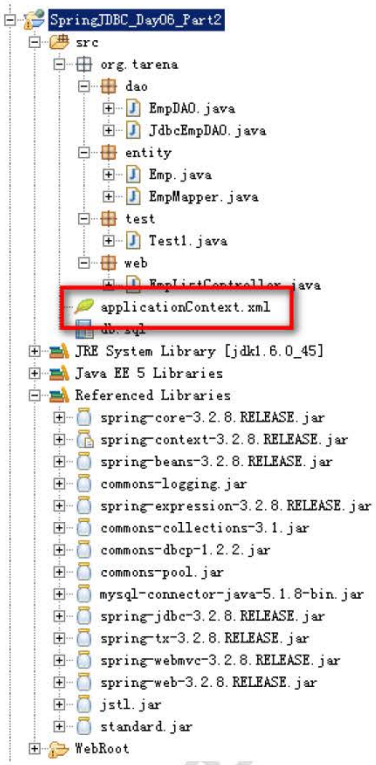


图-13

applicationContext.xml 文件中的代码如下所示：

```
    <?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:context="http://www.springframework.org/schema/context"
     xmlns:jdbc="http://www.springframework.org/schema/jdbc"
     xmlns:jee="http://www.springframework.org/schema/jee"
     xmlns:tx="http://www.springframework.org/schema/tx"
     xmlns:jpa="http://www.springframework.org/schema/data/jpa"
     xmlns:mvc="http://www.springframework.org/schema/mvc"
     xsi:schemaLocation="
         http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
         http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
         http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
         http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
         http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
         http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd
         http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

     <bean id="myDataSource"
         class="org.apache.commons.dbcp.BasicDataSource"
         destroy-method="close">
         <property name="driverClassName" value="com.mysql.jdbc.Driver" />
```

```
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="root" />
    </bean>

    <bean id="jdbcTemplate"
        class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="myDataSource"></property>
    </bean>

   </beans>
```

步骤四：新建 Emp 实体类和 EmpMapper 映射类

新建 Emp 实体类和 EmpMapper 映射类，Emp 和 EmpMapper 类与 SpringJDBC_Day06_Part1 练习的同名类完全一致，在此不再复述。

步骤五：新建 EmpDAO 接口

新建接口 EmpDAO，如下图-14 所示：

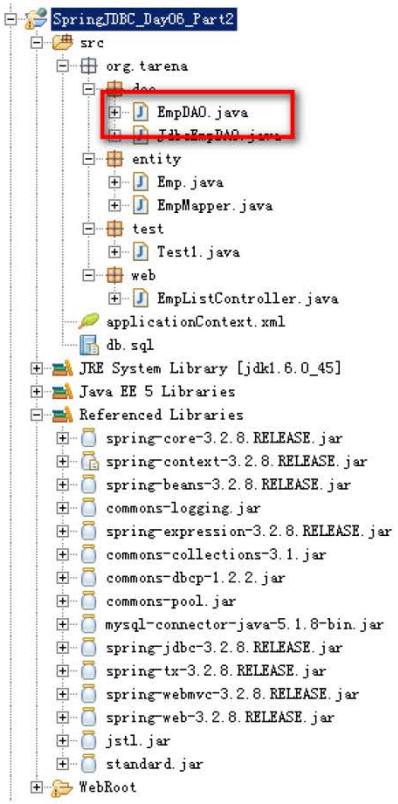


图-14

EmpDAO 接口的代码如下：

```
   package org.tarena.dao;

   import java.util.List;

   import org.tarena.entity.Emp;

   public interface EmpDAO {
    public List<Emp> findAll();
   }
```

步骤六：新建 JdbcEmpDAO 类

新建类 JdbcEmpDAO，如图-15 所示：

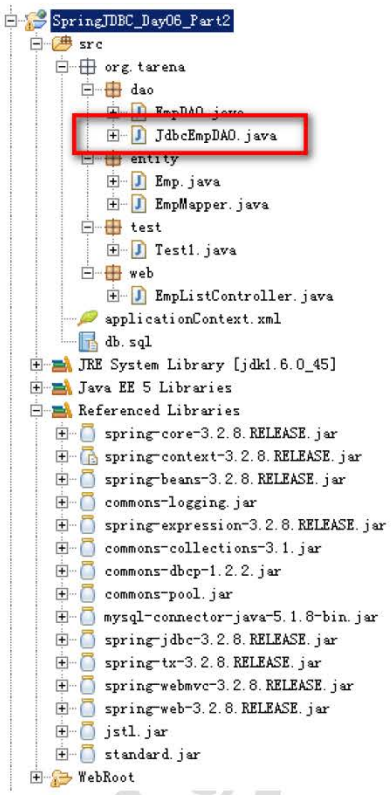


图-15

JdbcEmpDAO 类的代码如下：

```
package org.tarena.dao;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;
import org.tarena.entity.Emp;
import org.tarena.entity.EmpMapper;

@Repository
public class JdbcEmpDAO implements EmpDAO{

 private JdbcTemplate template;
 @Autowired
 public void setTemplate(JdbcTemplate template){
     this.template = template;
 }

 public List<Emp> findAll() {
     String sql = "select * from t_emp";
     RowMapper<Emp> mapper = new EmpMapper();
     List<Emp> list = template.query(sql, mapper);
     return list;
 }
}
```

新建 Test1 类，如下图-16 所示：

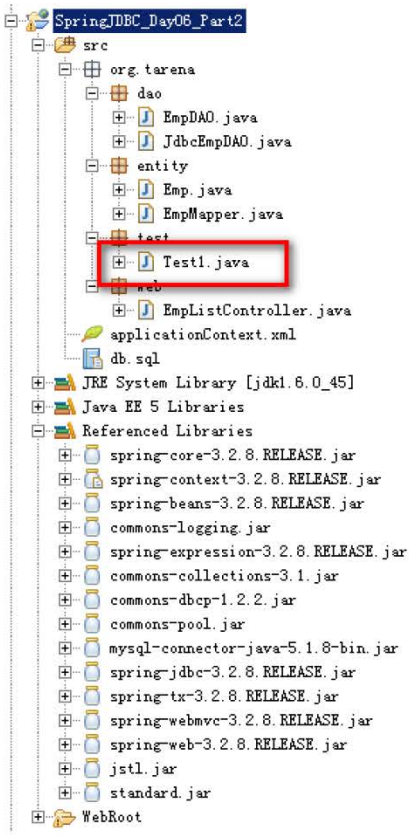


图-16

Test1 类代码如下：

```
package org.tarena.test;

import java.util.List;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

public class Test1 {
 /**
  * @param args
  */
 public static void main(String[] args) throws Exception{
     String conf = "applicationContext.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     EmpDAO empDao = ac.getBean("jdbcEmpDAO",EmpDAO.class);
     List<Emp> list = empDao.findAll();
     for(Emp emp : list){
         System.out.println(emp.getEmpno() +" "+emp.getEname());
     }
 }
}
```

步骤七：运行 Test1 类

运行 Test1 类，控制台输入结果如下图-17 所示：

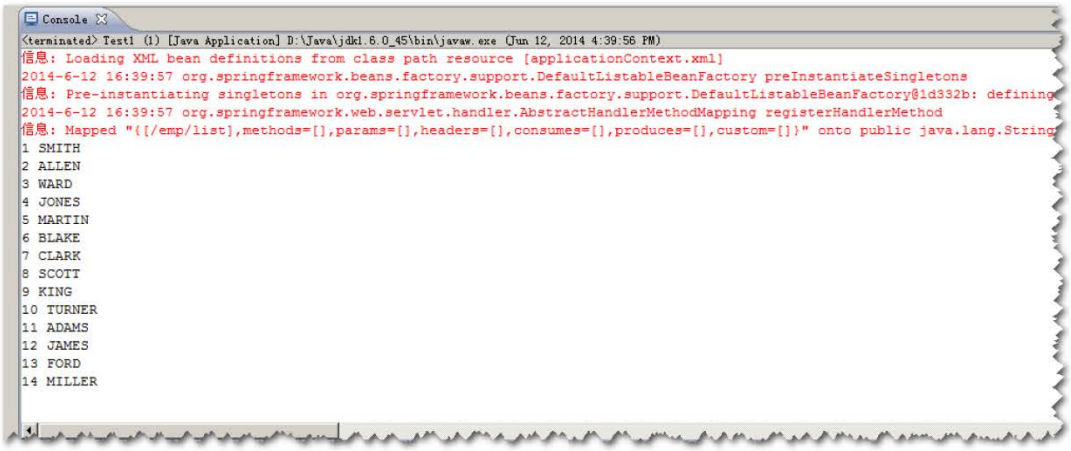


图-17

控制台输出如上图所示的信息，说明 JDBC Template 的 DAO 实现测试成功。

步骤八：创建 EmpListController 类

创建类 EmpListController，如图-18 所示：

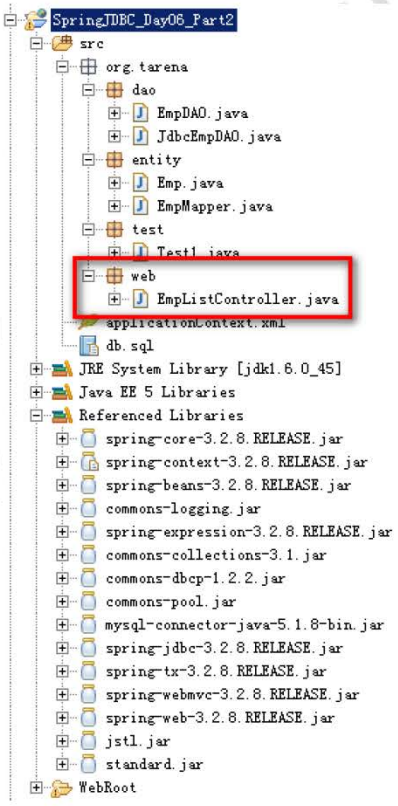


图-18

EmpListController 类代码如下：

```
package org.tarena.web;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

@Controller
public class EmpListController {
 private EmpDAO dao;
 @Autowired
 public void setDao(EmpDAO dao) {
     this.dao = dao;
 }
 @RequestMapping("/emp/list")
 public String execute(Model model){
     List<Emp> list = dao.findAll();
     model.addAttribute("emps", list);
     return "emp_list";
 }
}
```

步骤九：修改 web.xml 文件

修改 web.xml，加入方框部分 XML 配置信息，如图-19 所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <servlet>
        <servlet-name>SpringMVC</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:applicationContext.xml</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>SpringMVC</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

图-19

XML 配置如下：

```
    <servlet>
        <servlet-name>SpringMVC</servlet-name>

    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet
-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:applicationContext.xml</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>SpringMVC</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
```

步骤十：创建 emp_list.jsp 文件

创建 emp_list.jsp，如图-20 所示：

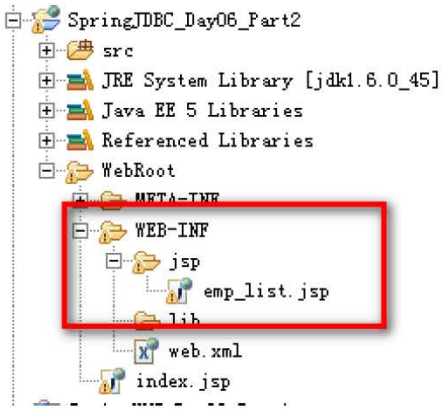


图-20

emp_list.jsp 代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
  <head>
    <title>员工列表示例</title>
  </head>

  <body>
    <table border="1">
      <tr>
          <td>编号</td>
          <td>姓名</td>
          <td>工资</td>
          <td>入职时间</td>
      </tr>
      <c:forEach items="${emps}" var="emp">
      <tr>
         <td>${emp.empno }</td>
          <td>${emp.ename }</td>
          <td>${emp.sal }</td>
          <td>${emp.hiredate }</td>
      </tr>
      </c:forEach>
    </table>
  </body>
</html>
```

步骤十一：修改 applicationContext.xml 文件

修改 applicationContext.xml，加入方框部分 XML 配置信息，如图-21 所示：

```
    <bean id="myDataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="root" />
    </bean>

    <bean id="jdbcTemplate"
        class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="myDataSource"></property>
    </bean>

    <context:component-scan base-package="org.tarena" />

    <!-- 支持@RequestMapping请求和Controller映射 -->
    <mvc:annotation-driven />

    <!-- SpringMVC的ViewResolver -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property value="/WEB-INF/jsp/" name="prefix" />
        <property value=".jsp" name="suffix" />
    </bean>
</beans>
```

图-21

XML 配置如下：

```
    <context:component-scan base-package="org.tarena" />

    <!-- 支持@RequestMapping 请求和 Controller 映射 -->
    <mvc:annotation-driven />

    <!-- SpringMVC 的 ViewResolver -->
    <bean

    class="org.springframework.web.servlet.view.InternalResourceViewResolver"
>
        <property value="/WEB-INF/jsp/" name="prefix" />
        <property value=".jsp" name="suffix" />
    </bean>
```

步骤十二：以 Web 方式运行项目

以 Web 方式运行本项目，如下图-22 所示：

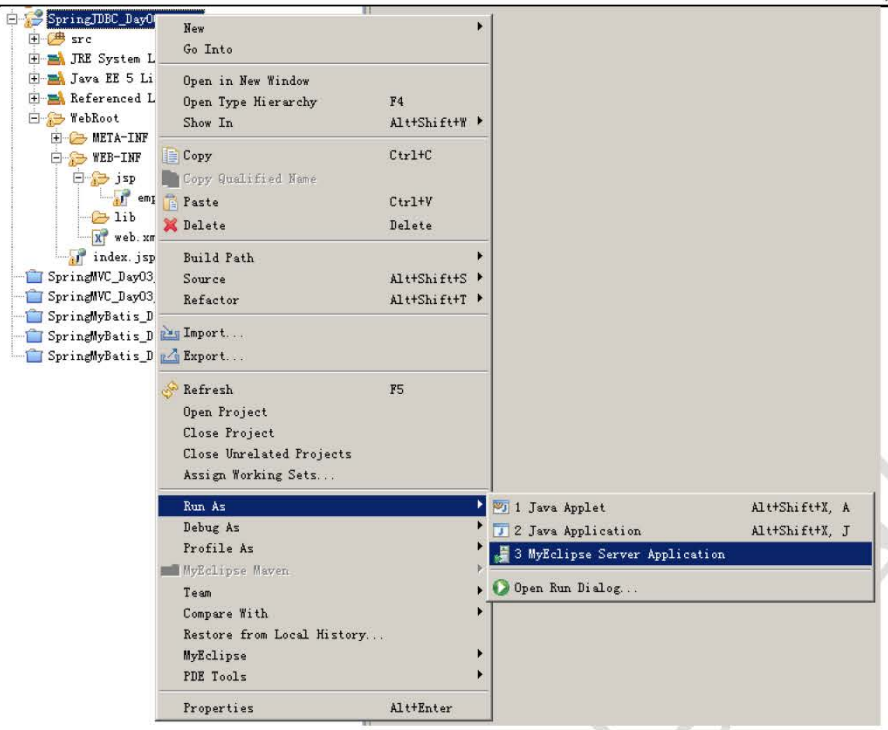
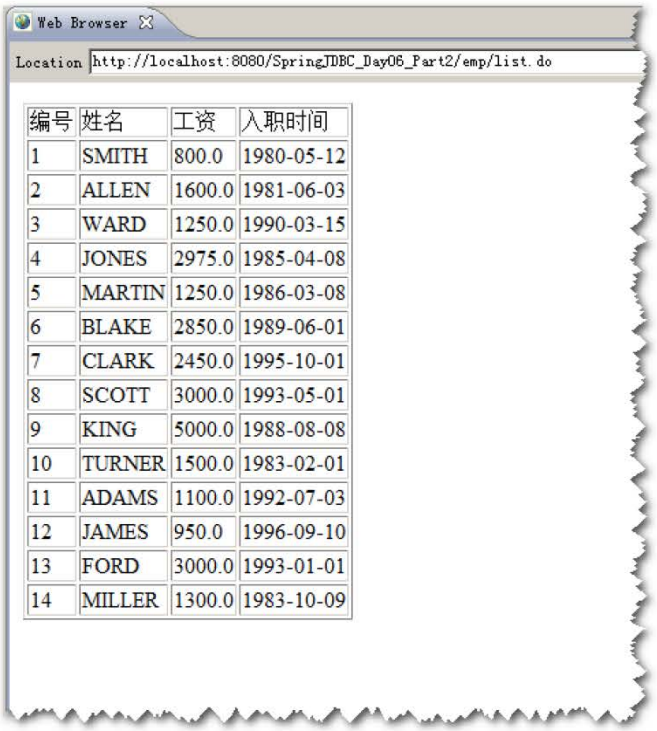


图-22

在浏览器输入地址：
http://localhost:8080/SpringJDBC_Day06_Part2/emp/list.do
查看运行结果。



编号	姓名	工资	入职时间
1	SMITH	800.0	1980-05-12
2	ALLEN	1600.0	1981-06-03
3	WARD	1250.0	1990-03-15
4	JONES	2975.0	1985-04-08
5	MARTIN	1250.0	1986-03-08
6	BLAKE	2850.0	1989-06-01
7	CLARK	2450.0	1995-10-01
8	SCOTT	3000.0	1993-05-01
9	KING	5000.0	1988-08-08
10	TURNER	1500.0	1983-02-01
11	ADAMS	1100.0	1992-07-03
12	JAMES	950.0	1996-09-10
13	FORD	3000.0	1993-01-01
14	MILLER	1300.0	1983-10-09

图-23

浏览器显示如上图-23 所示的信息，说明 Spring+JDBC TemplateWeb 功能测试成功。

- **完整代码**

类 org.tarena.entity.Emp 代码如下：

```
package org.tarena.entity;
import java.sql.Date;

public class Emp {
 private Integer empno;
 private String ename;
 private String job;
 private Integer mgr;
 private Date hiredate;
 private Double sal;
 private Double comm;
 private Integer deptno;

 public Integer getEmpno() {
     return empno;
 }
 public void setEmpno(Integer empno) {
     this.empno = empno;
 }
 public String getEname() {
     return ename;
 }
 public void setEname(String ename) {
     this.ename = ename;
 }
 public String getJob() {
     return job;
 }
 public void setJob(String job) {
     this.job = job;
 }
 public Integer getMgr() {
     return mgr;
 }
 public void setMgr(Integer mgr) {
     this.mgr = mgr;
 }
 public Date getHiredate() {
     return hiredate;
 }
 public void setHiredate(Date hiredate) {
     this.hiredate = hiredate;
 }
 public Double getSal() {
     return sal;
 }
 public void setSal(Double sal) {
     this.sal = sal;
 }
 public Double getComm() {
     return comm;
 }
 public void setComm(Double comm) {
     this.comm = comm;
 }
 public Integer getDeptno() {
     return deptno;
 }
 public void setDeptno(Integer deptno) {
     this.deptno = deptno;
 }
}
```

类 org.tarena.entity.EmpMapper 代码如下：

```
package org.tarena.entity;

import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

public class EmpMapper implements RowMapper<Emp>{
 public Emp mapRow(ResultSet rs, int arg1) throws SQLException {
     Emp emp = new Emp();
     emp.setEmpno(rs.getInt("EMPNO"));
     emp.setEname(rs.getString("ENAME"));
     emp.setJob(rs.getString("JOB"));
     emp.setMgr(rs.getInt("MGR"));
     emp.setHiredate(rs.getDate("HIREDATE"));
     emp.setSal(rs.getDouble("SAL"));
     emp.setComm(rs.getDouble("COMM"));
     emp.setDeptno(rs.getInt("DEPTNO"));
     return emp;
 }
}
```

接口 org.tarena.dao.EmpDAO 代码如下：

```
package org.tarena.dao;

import java.util.List;
import org.tarena.entity.Emp;

public interface EmpDAO {
 public List<Emp> findAll();
}
```

类 org.tarena.dao.JdbcEmpDAO 代码如下：

```
package org.tarena.dao;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;
import org.tarena.entity.Emp;
import org.tarena.entity.EmpMapper;

@Repository
public class JdbcEmpDAO implements EmpDAO{

 private JdbcTemplate template;
 @Autowired
 public void setTemplate(JdbcTemplate template){
     this.template = template;
 }

 public List<Emp> findAll() {
     String sql = "select * from t_emp";
     RowMapper<Emp> mapper = new EmpMapper();
     List<Emp> list = template.query(sql, mapper);
     return list;
 }
```

类 org.tarena.test.Test1 代码如下：

```
package org.tarena.test;

import java.util.List;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

public class Test1 {
 /**
  * @param args
  */
 public static void main(String[] args) throws Exception{
     String conf = "applicationContext.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     EmpDAO empDao = ac.getBean("jdbcEmpDAO",EmpDAO.class);
     List<Emp> list = empDao.findAll();
     for(Emp emp : list){
         System.out.println(emp.getEmpno() +" "+emp.getEname());
     }
 }
}
```

类 org.tarena.web.EmpListController 代码如下：

```
package org.tarena.web;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

@Controller
public class EmpListController {
 private EmpDAO dao;
 @Autowired
 public void setDao(EmpDAO dao) {
     this.dao = dao;
 }
 @RequestMapping("/emp/list")
 public String execute(Model model){
     List<Emp> list = dao.findAll();
     model.addAttribute("emps", list);
     return "emp list";
 }
}
```

Spring 配置文件 applicationContext.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:jdbc="http://www.springframework.org/schema/jdbc"
 xmlns:jee="http://www.springframework.org/schema/jee"
```

```
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd
        http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

    <bean id="myDataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="root" />
    </bean>

    <bean id="jdbcTemplate"
        class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="myDataSource"></property>
    </bean>

    <context:component-scan base-package="org.tarena" />

    <!-- 支持@RequestMapping 请求和 Controller 映射 -->
    <mvc:annotation-driven />

    <!-- SpringMVC 的 ViewResolver -->
    <bean

    class="org.springframework.web.servlet.view.InternalResourceViewResolver"
>
        <property value="/WEB-INF/jsp/" name="prefix" />
        <property value=".jsp" name="suffix" />
    </bean>
   </beans>
```

Jsp 页面 emp_list.jsp 代码如下：

```
   <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
   <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
   <html>
     <head>
       <title>员工列表示例</title>
     </head>

     <body>
       <table border="1">
         <tr>
             <td>编号</td>
             <td>姓名</td>
             <td>工资</td>
             <td>入职时间</td>
         </tr>
```

```
        <c:forEach items="${emps}" var="emp">
        <tr>
            <td>${emp.empno }</td>
             <td>${emp.ename }</td>
             <td>${emp.sal }</td>
             <td>${emp.hiredate }</td>
        </tr>
        </c:forEach>
       </table>
     </body>
   </html>
```

课后作业

1. 利用 Spring+JDBC Template 实现员工更新功能（基于 T_EMP 表）。

达内IT培训集团

MyBatis 核心

Unit02

知识体系.....Page 43

MyBatis 基础	MyBatis 框架简介	什么是 MyBatis
		MyBatis 体系结构
		MyBatis 配置文件
		框架 API 简介
	MyBatis 基本应用	搭建 MyBatis 技术环境
		获取 SqlSession 对象
		利用 SqlSession 实现 CRUD 操作
		利用 MyBatis 实现分页查询
		返回 Map 类型查询结果
		使用 Mapper 映射器
		ResultMap 映射定义

经典案例.....Page 51

获取 SqlSession 对象案例	获取 SqlSession 对象
实现对 Dept 表的 CRUD 操作案例	利用 SqlSession 实现 CRUD 操作
实现对 Dept 表的分页查询操作案例	利用 MyBatis 实现分页查询
实现对 Dept 表的 Map 类型查询操作案例	返回 Map 类型查询结果
使用 Mapper 对 Dept 表操作案例	使用 Mapper 映射器
利用 ResultMap 自定义映射案例	ResultMap 映射定义

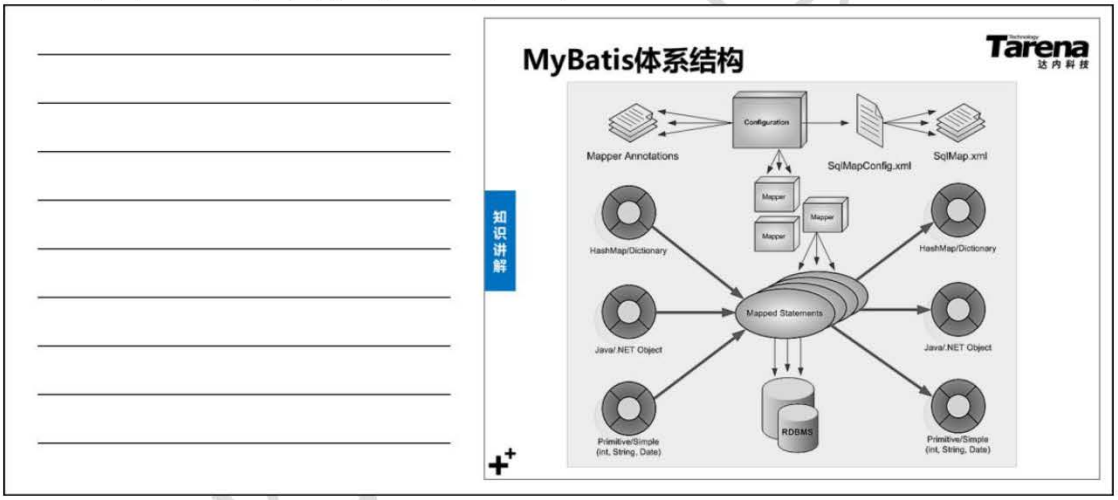
课后作业.....Page 100

1. MyBatis 基础

1.1. MyBatis 框架简介

1.1.1. 【MyBatis 框架简介】什么是 MyBatis

1.1.2. 【MyBatis 框架简介】MyBatis 体系结构



MyBatis体系结构（续2）

- 3）SQL执行
  - 将最终得到的SQL和参数拿到数据库进行执行，得到操作数据库的结果。
- 4）结果映射
  - 将操作数据库的结果按照映射的配置进行转换，可以转换成HashMap、JavaBean或者基本数据类型，并将最终结果返回。

知识讲解

1.1.3. 【MyBatis 框架简介】MyBatis 配置文件

MyBatis配置文件

MyBatis框架的XML配置文件包含下面两种类型

- 1）SqlMapConfig.xml (1个)
  - 主配置文件，用于指定数据库连接参数和框架参数
- 2）SqlMap.xml (n个)
  - 映射定义文件，用于定义SQL语句和映射信息

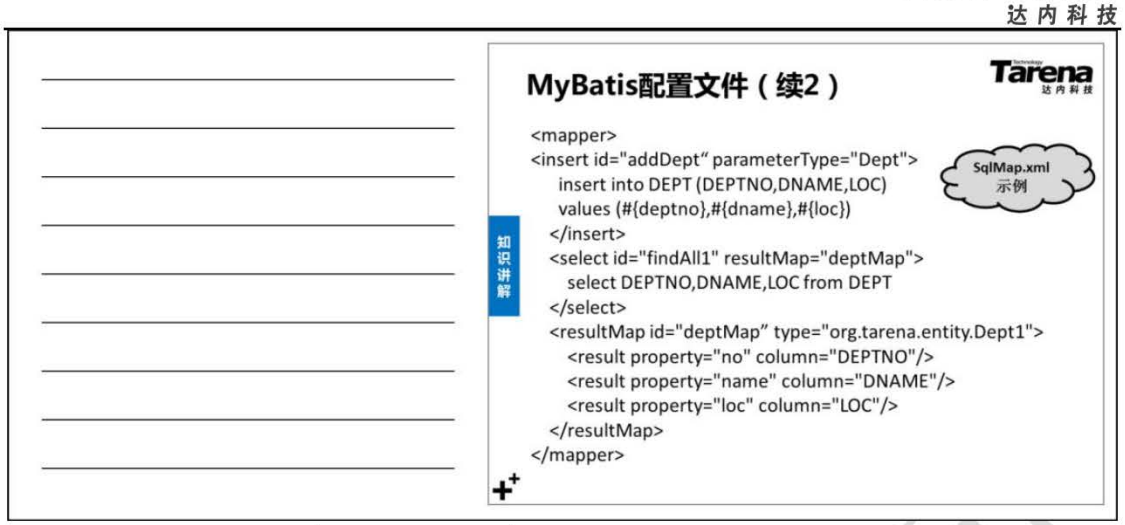
知识讲解

MyBatis配置文件（续1）

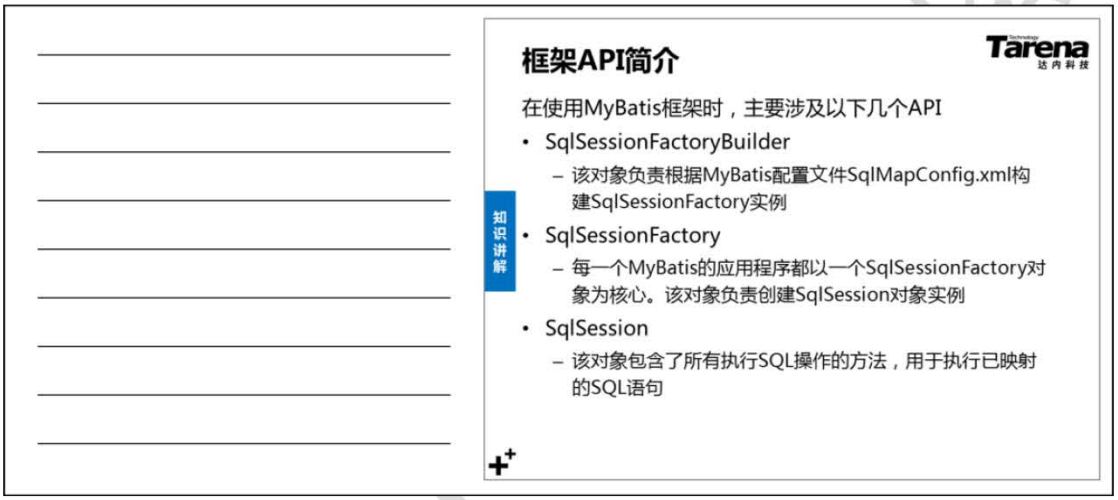
SqlMapConfig.xml 示例

```
<configuration>
<environments default="environment">
  <environment id="environment">
    <transactionManager type="JDBC" />
    <dataSource type="POOLED">
      <property name="driver"
        value="oracle.jdbc.OracleDriver"/>
      <property name="url"
        value="jdbc:oracle:thin:@localhost:tarena10g" />
      <property name="username" value="demo" />
      <property name="password" value="demo" />
    </dataSource>
  </environment>
</environments>
</configuration>
```

知识讲解



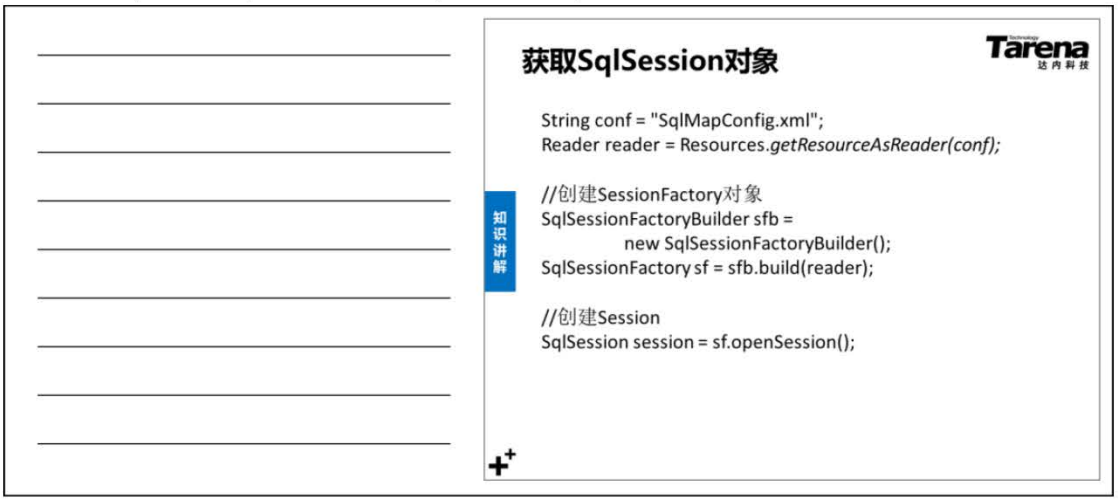
1.1.4. 【MyBatis 框架简介】框架 API 简介



1.2. MyBatis 基本应用

1.2.1. 【MyBatis 基本应用】搭建 MyBatis 技术环境

1.2.2. 【MyBatis 基本应用】获取 SqlSession 对象



1.2.3. 【MyBatis 基本应用】利用 SqlSession 实现 CRUD 操作

利用SqlSession实现CRUD操作（续2）

解决问题

- 更新操作的SqlMap.xml定义

```
<update id="updateDept"
    parameterType="org.tarena.entity.Dept">
    update DEPT set DNAME=#{dname},LOC=#{loc}
    where DEPTNO=#{deptno}
</update>
```

- 更新操作的Java代码

```
SqlSession session = sf.openSession();
//调用updateDept操作
session.update("updateDept",dept);
session.commit();
Session.close();
```

利用SqlSession实现CRUD操作（续3）

解决问题

- 删除操作的SqlMap.xml定义

```
<delete id="deleteById" parameterType="int">
    delete from DEPT where DEPTNO=#{no}
</delete>
```

- 删除操作的Java代码

```
SqlSession session = sf.openSession();
//调用deleteDept操作
session.delete("deleteById",10);
session.commit();
Session.close();
```

利用SqlSession实现CRUD操作（续4）

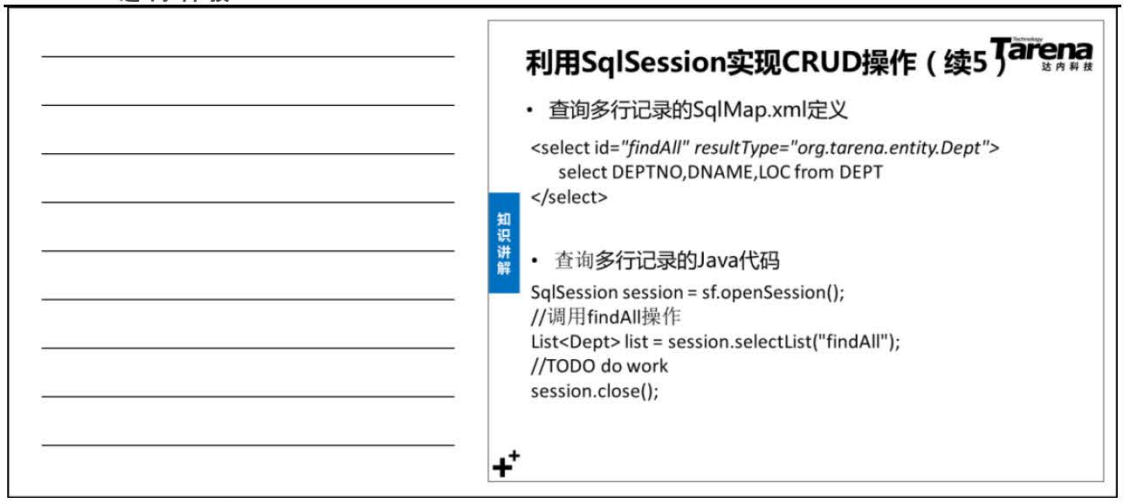
解决问题

- 查询单行记录的SqlMap.xml定义

```
<select id="findById"
        parameterType="int"
        resultType="org.tarena.entity.Dept">
    select DEPTNO,DNAME,LOC from DEPT where DEPTNO=#{id}
</select>
```

- 查询单行记录的Java代码

```
SqlSession session = sf.openSession();
//调用findById操作
Dept dept = (Dept)session.selectOne("findById",10);
//TODO do work
session.close();
```

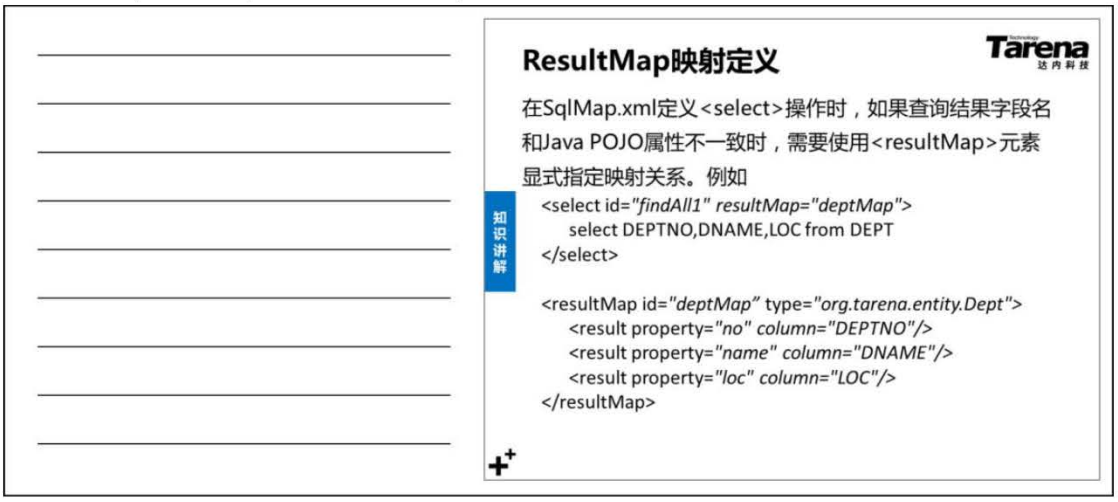


1.2.4. 【MyBatis 基本应用】利用 MyBatis 实现分页查询

1.2.5. 【MyBatis 基本应用】返回 Map 类型查询结果

1.2.6. 【MyBatis 基本应用】使用 Mapper 映射器

1.2.7. 【MyBatis 基本应用】ResultMap 映射定义



经典案例

1. 获取 SqlSession 对象案例

- **问题**

在 MyBaits 中，对数据库操作首先要先获取 SqlSession 对象，那么该如何获取呢？

- **方案**

首先用 SqlSessionFactoryBuilder 获取 SqlSessionFactory 实例，然后通过 SqlSessionFactory 再获取 SqlSession 对象。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：在 MySQL 的 test 数据库中创建表和数据

请在 MySQL 的 test 数据库中执行以下 SQL 语句，用来创建表和相关数据：

```
create table t_dept(
  deptno int primary key,
  dname varchar(20),
  loc varchar(50)
);

insert into t_dept values (10,'testing','beijing');
insert into t_dept values (20,'saling','shanghai');
```

步骤二：新建工程，导入 jar 包

新建名为 SpringMyBatis_Day07 的 Java 工程,在该工程导入如图-1 所示的 jar 包：

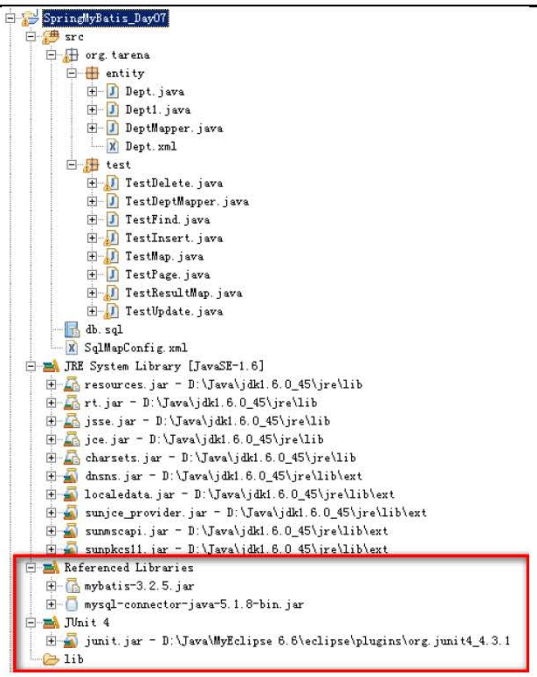


图-1

步骤三：新建 SqlMapConfig.xml 文件

新建 SqlMapConfig.xml 文件，如图-2 所示：

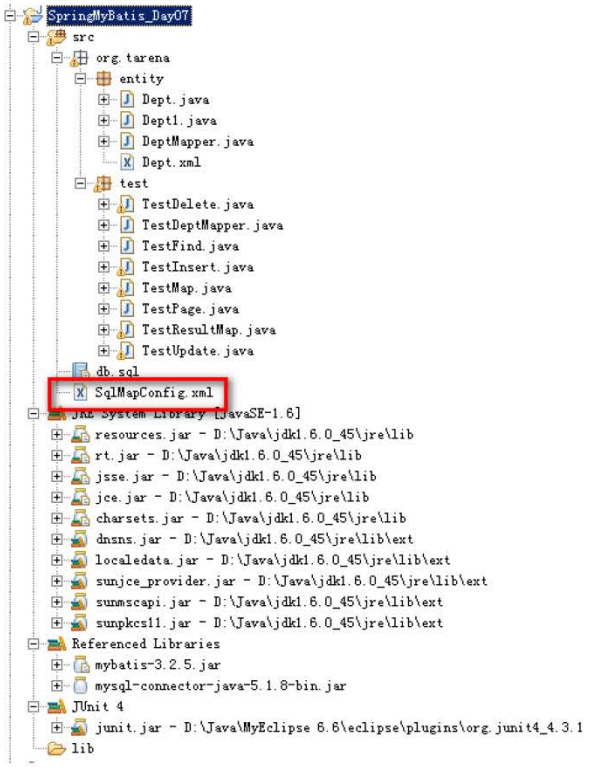


图-2

SqlMapConfig.xml 文件中的代码如下所示：

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

```
<!DOCTYPE configuration PUBLIC "-//ibatis.apache.org//DTD Config 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-config.dtd">
<configuration>
 <environments default="environment">
     <environment id="environment">
         <transactionManager type="JDBC" />
         <dataSource type="POOLED">
             <property name="driver" value="com.mysql.jdbc.Driver" />
             <property name="url"
                 value="jdbc:mysql://localhost:3306/test" />
             <property name="username" value="root" />
             <property name="password" value="root" />
         </dataSource>
     </environment>
 </environments>
</configuration>
```

步骤四：新建 TestSqlSession 测试类

新建类 TestSqlSession，如图-3 所示：

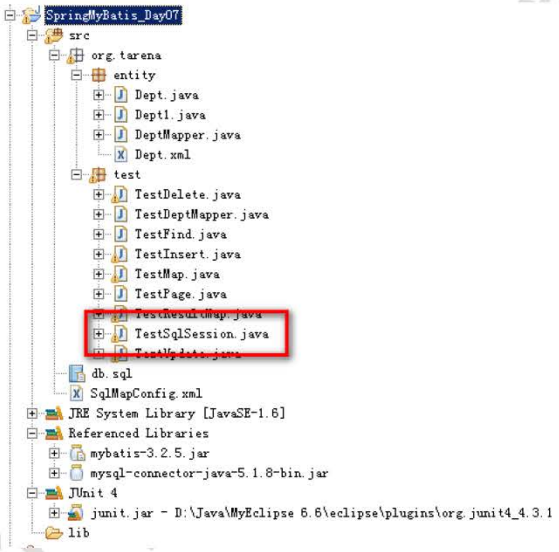


图-3

类 TestSqlSession 中的代码如下所示：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestSqlSession {
 @Test
 public void testSqlSession() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
```

```
        SqlSessionFactory sf = sfb.build(reader);
        //创建 Session
        SqlSession session = sf.openSession();
        System.out.println(session);

        //关闭 Session
        session.close();
    }
}
```

步骤五：运行 TestSqlSession 测试

在 TestSqlSession 右键运行测试，如图-4 所示：

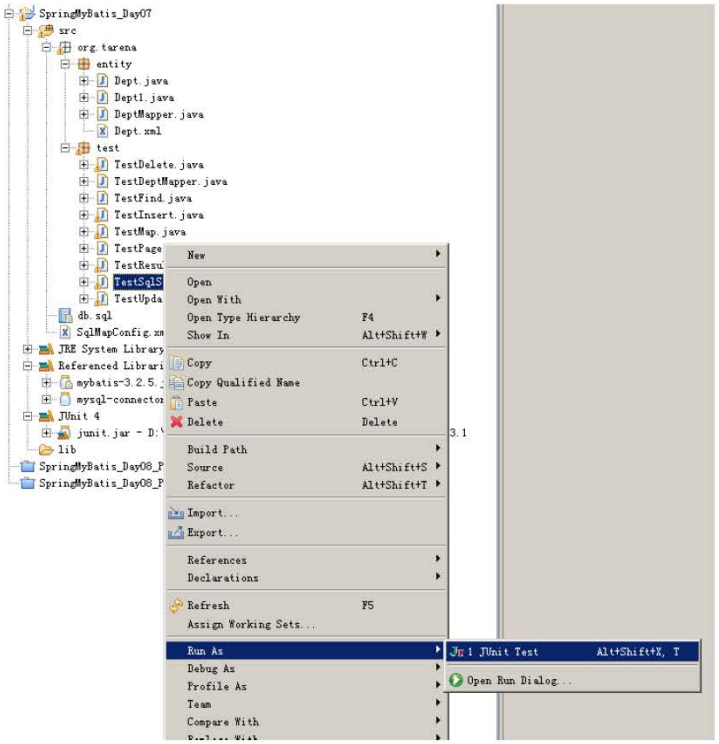


图-4

控制台输出如图-5 所示的信息，说明测试成功。

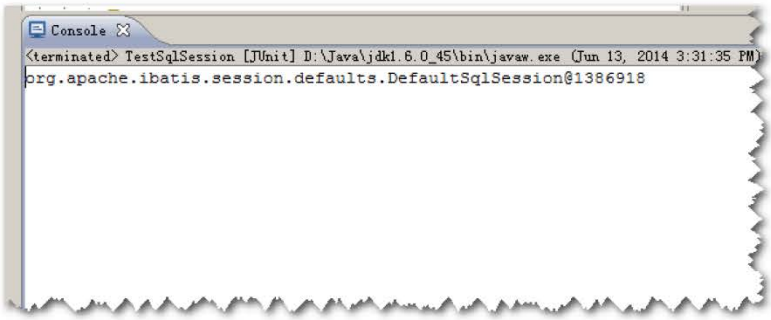


图-5

- **完整代码**

我们使用以下数据库结构：

```
create table t_dept(
  deptno int primary key,
  dname varchar(20),
  loc varchar(50)
);

insert into t_dept values (10,'testing','beijing');
insert into t_dept values (20,'saling','shanghai');
```

MyBatis 配置文件 SqlMapConfig.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration PUBLIC "-//ibatis.apache.org//DTD Config 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-config.dtd">
<configuration>
 <environments default="environment">
     <environment id="environment">
         <transactionManager type="JDBC" />
         <dataSource type="POOLED">
             <property name="driver" value="com.mysql.jdbc.Driver" />
             <property name="url"
                 value="jdbc:mysql://localhost:3306/test" />
             <property name="username" value="root" />
             <property name="password" value="root" />
         </dataSource>
     </environment>
 </environments>
</configuration>
```

测试类 TestSqlSession 代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestSqlSession {
 @Test
 public void testSqlSession() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     System.out.println(session);
     //关闭 Session
     session.close();
 }
}
```

2. 实现对 Dept 表的 CRUD 操作案例

- **问题**

如何使用 SqlSession 对象数据表进行增加、删除、修改和查询操作？

- **方案**

首先根据数据表编写实体类，然后编写 Mapper 映射文件，将 SQL 语句写在 Mapper 映射文件中，最后利用 SqlSession 对象提供的方法进行操作。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：新建 Dept 实体类

新建类 Dept，如图-6 所示：

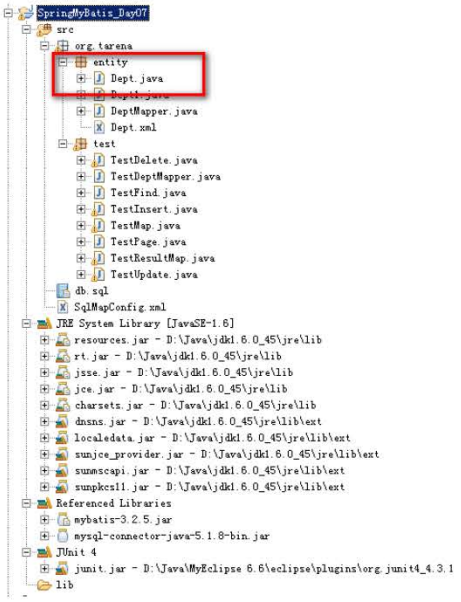


图-6

Dept 类文件中的代码如下所示：

```
package org.tarena.entity;

public class Dept{
 private Integer deptno;
 private String dname;
 private String loc;
 public Integer getDeptno() {
     return deptno;
 }
 public void setDeptno(Integer deptno) {
     this.deptno = deptno;
 }
 public String getDname() {
```

```
        return dname;
    }
    public void setDname(String dname) {
        this.dname = dname;
    }
    public String getLoc() {
        return loc;
    }
    public void setLoc(String loc) {
        this.loc = loc;
    }
}
```

步骤二：新建 DeptMapper 类和 Dept.xml

新建 DeptMapper 类和 Dept.xml 映射文件，如下图-7 所示：

图-7

DeptMapper 接口代码如下：

```
package org.tarena.entity;
import java.util.List;

public interface DeptMapper {
 public void addDept(Dept dept);
}
```

Dept.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">
<insert id="addDept"
 parameterType="org.tarena.entity.Dept">
insert into T_DEPT (DEPTNO,DNAME,LOC)
 values (#{deptno},#{dname},#{loc})
</insert>
```

```
</mapper>
```

步骤三：修改 SqlMapConfig.xml 文件

修改 SqlMapConfig.xml 文件，加入图-8 方框中的配置：

图-8

SqlMapConfig.xml 文件加入的代码如下：

```
<mappers>
    <mapper resource="org/tarena/entity/Dept.xml" />
</mappers>
```

步骤四：新建 TestInsert 类

新建类 TestInsert，如图-9 所示：

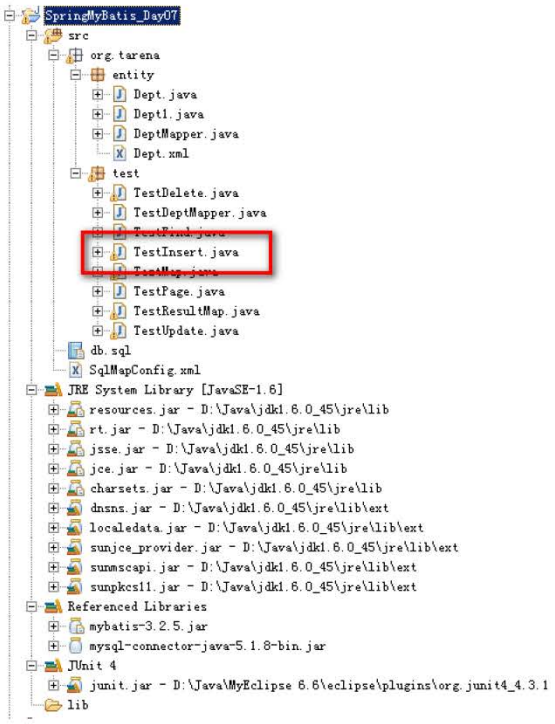


图-9

TestInsert 类的代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.RowBounds;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestInsert {

 @Test
 public void testInsert() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader =
         Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb =
             new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     //调用 addDept 操作
     Dept dept = new Dept();
     dept.setDeptno(60);
     dept.setDname("testing");
     dept.setLoc("beijing");
     session.insert("addDept",dept);
     session.commit();
     //关闭
     session.close();
 }
}
```

步骤五：运行 TestInsert 测试

在 TestInsert 类右键运行测试，如图-10 所示：

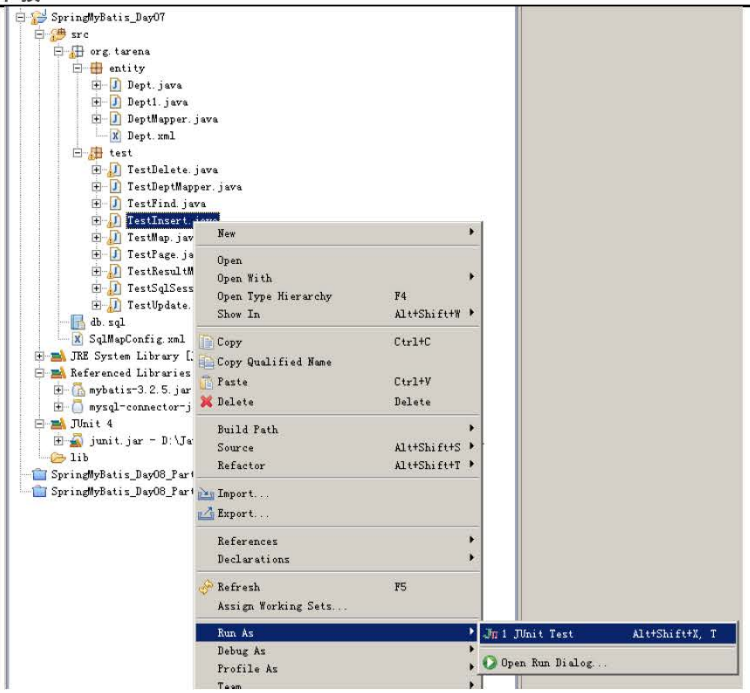


图-10

查看数据库表，新增加一条记录说明测试成功。如图-11 所示：

deptno	dname	loc
10	testing	beijing
20	saling	shanghai
60	testing	beijing

图-11

步骤六：修改 DeptMapper 接口和 Dept.xml

DeptMapper 接口加入下图-12 方框内的代码：

```
package org.tarena.entity;

import java.util.List;

public interface DeptMapper {
    public void addDept(Dept dept);

    public void updateDept(Dept dept);

}
```

图-12

DeptMapper 接口加入的代码如下：

```
    public void updateDept(Dept dept);
```

Dept.xml 映射文件加入如图-13 方框内的配置：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">
    <insert id="addDept" parameterType="org.tarena.entity.Dept">
        insert into T_DEPT (DEPTNO,DNAME,LOC) values
        (#{deptno},#{dname},#{loc})
    </insert>

    <update id="updateDept" parameterType="org.tarena.entity.Dept">
        update T_DEPT set DNAME=#{dname},LOC=#{loc} where
        DEPTNO=#{deptno}
    </update>

</mapper>
```

图-13

Dept.xml 映射文件加入的代码如下：

```
<update id="updateDept" parameterType="org.tarena.entity.Dept">
    update T_DEPT set DNAME=#{dname},LOC=#{loc} where
    DEPTNO=#{deptno}
</update>
```

步骤七：新建 TestUpdate 类

新建类 TestUpdate，如图-14 所示：

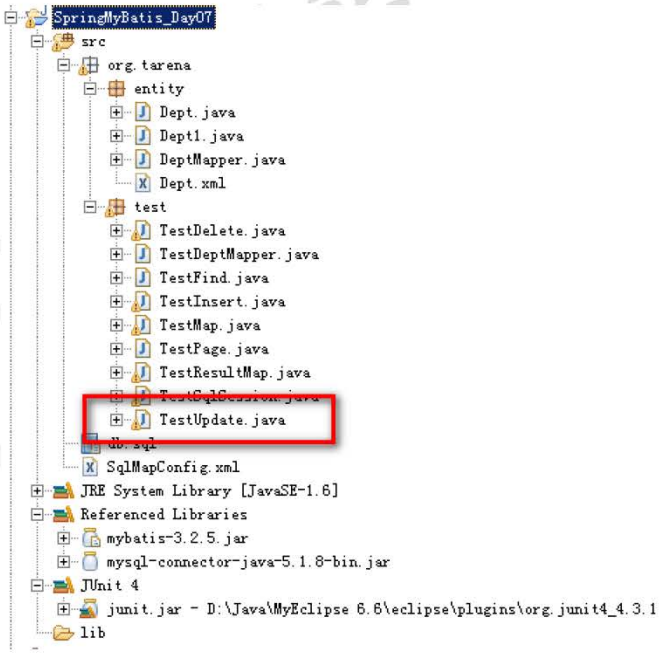


图-14

TestUpdate 类的代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
```

```
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.RowBounds;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestUpdate {

 @Test
 public void testUpdate() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader =
         Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb =
             new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     //调用 findById 操作
     Dept dept = (Dept)
         session.selectOne("findById",60);
     dept.setDname("开发部");
     dept.setLoc("北京");
     //调用 updateDept 操作
     session.update("updateDept",dept);
     session.commit();
     //关闭
     session.close();
 }

}
```

步骤八：运行 TestUpdate 测试

在 TestUpdate 类右键运行测试，如图-15 所示：

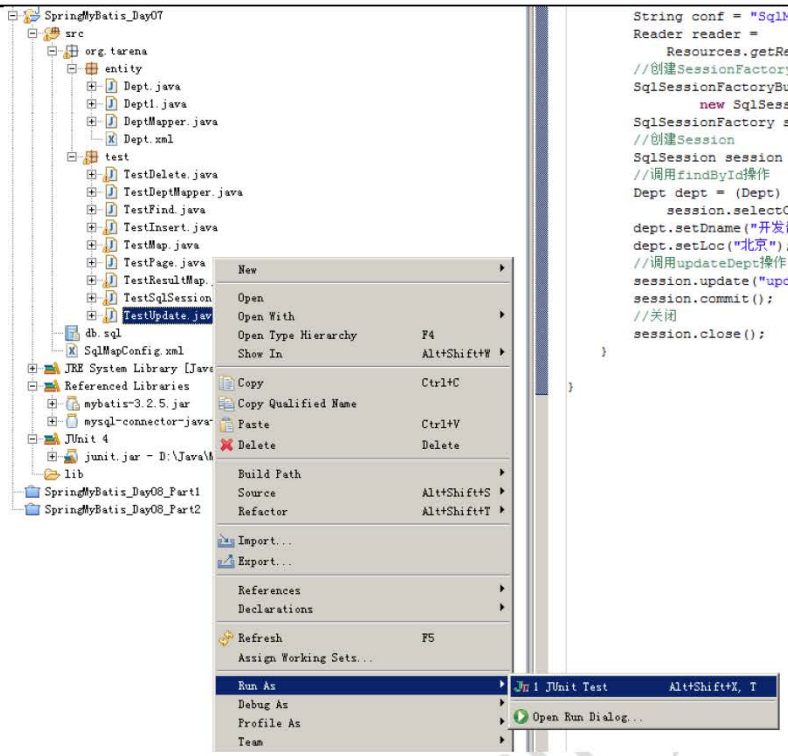


图-15

查看数据库表，deptno 为 60 的记录修改成功，说明测试成功。如图-16 所示：

deptno	dname	loc
10	testing	beijing
20	seling	shanghai
60	开发部	北京

图-16

步骤九：修改 DeptMapper 接口和 Dept.xml

DeptMapper 接口加入下图-17 方框内的代码：

```
package org.tarena.entity;

import java.util.List;

public interface DeptMapper {
    public void addDept(Dept dept);

    public void updateDept(Dept dept);

    public void deleteById(int id);

}
```

图-17

DeptMapper 接口加入的代码如下：

```
public void deleteById(int id);
```

Dept.xml 映射文件加入下图-18 方框内的配置：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">
    <insert id="addDept" parameterType="org.tarena.entity.Dept">
        insert into T_DEPT (DEPTNO,DNAME,LOC) values
        (#{deptno},#{dname},#{loc})
    </insert>

    <update id="updateDept" parameterType="org.tarena.entity.Dept">
        update T_DEPT set DNAME=#{dname},LOC=#{loc} where
        DEPTNO=#{deptno}
    </update>

    <delete id="deleteById" parameterType="int">
        delete from T_DEPT where DEPTNO=#{no}
    </delete>

</mapper>
```

图-18

Dept.xml 映射文件加入的代码如下：

```
<delete id="deleteById" parameterType="int">
    delete from T_DEPT where DEPTNO=#{no}
</delete>
```

步骤十：新建 TestDelete 类

新建类 TestDelete，如图-19 下所示：

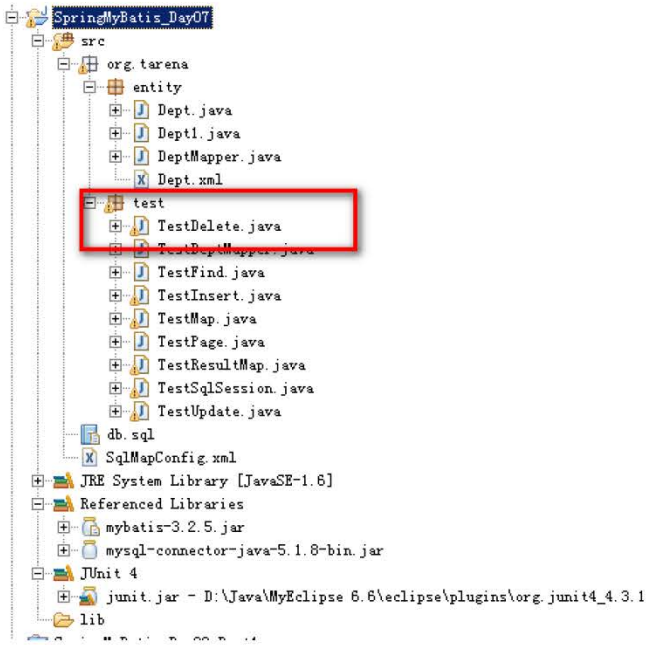


图-19

TestDelete 类的代码如下：

```
package org.tarena.test;
```

```
import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.RowBounds;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestDelete {
 @Test
 public void testDelete() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     //调用 deleteById 操作
     session.delete("deleteById",60);
     session.commit();
     //关闭
     session.close();
 }
}
```

步骤十一：运行 TestDelete 测试

在 TestDelete 类右键运行测试，如图-20 所示：

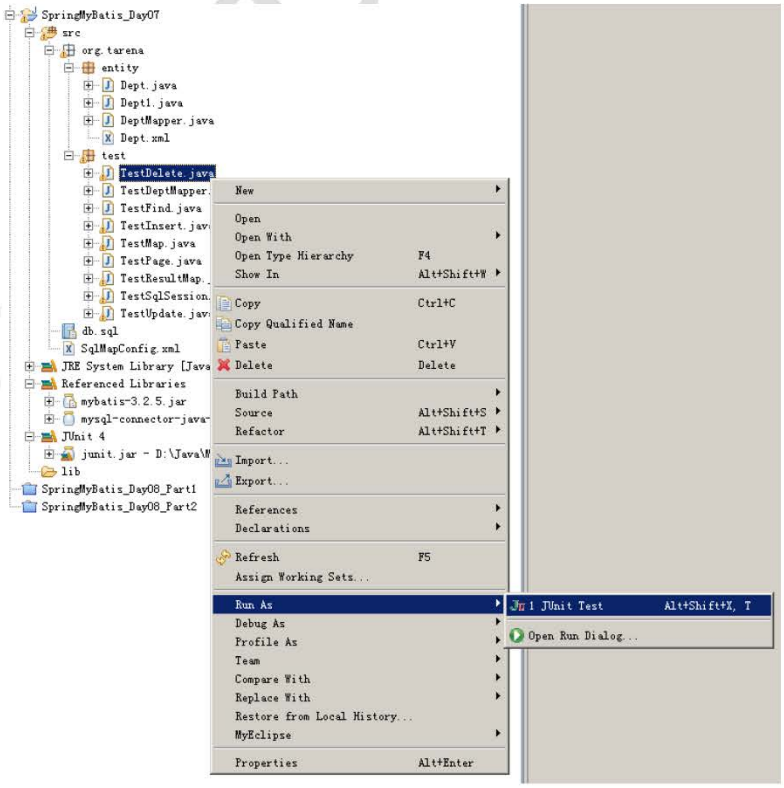


图-20

查看数据库表，deptno 为 60 的记录被删除，说明测试成功。

步骤十二：修改 DeptMapper 接口和 Dept.xml

DeptMapper 接口加入下图-21 方框内的代码：

```
package org.tarena.entity;

import java.util.List;

public interface DeptMapper {
    public void addDept(Dept dept);

    public void updateDept(Dept dept);

    public void deleteById(int id);

    public Dept findById(int id);
}
```

图-21

DeptMapper 接口加入的代码如下：

```
public Dept findById(int id);
```

Dept.xml 映射文件加入下图-22 方框内的配置：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">
    <insert id="addDept" parameterType="org.tarena.entity.Dept">
        insert into T_DEPT (DEPTNO,DNAME,LOC) values
        (#{deptno},#{dname},#{loc})
    </insert>

    <update id="updateDept" parameterType="org.tarena.entity.Dept">
        update T_DEPT set DNAME=#{dname},LOC=#{loc} where
        DEPTNO=#{deptno}
    </update>

    <delete id="deleteById" parameterType="int">
        delete from T_DEPT where DEPTNO=#{no}
    </delete>

    <select id="findById" parameterType="int"
        resultType="org.tarena.entity.Dept">
        select DEPTNO,DNAME,LOC from T_DEPT where DEPTNO=#{id}
    </select>

</mapper>
```

图-22

Dept.xml 映射文件加入的代码如下：

```
<select id="findById" parameterType="int"
    resultType="org.tarena.entity.Dept">
    select DEPTNO,DNAME,LOC from T_DEPT where DEPTNO=#{id}
</select>
```

步骤十三：新建 TestFind 类

新建类 TestFind，如图-23 所示：

图-23

TestFind 类的代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestFind {
 @Test
 public void testFindById() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     //调用 findById 方法
     Dept dept = (Dept)session.selectOne("findById",10);
     System.out.println(dept.getDeptno()+" "
             +dept.getDname()+" "
             +dept.getLoc());
     //关闭
     session.close();
 }
}
```

步骤十四：运行 TestFind 测试

在 TestFind 类右键运行测试，如图-24 所示：

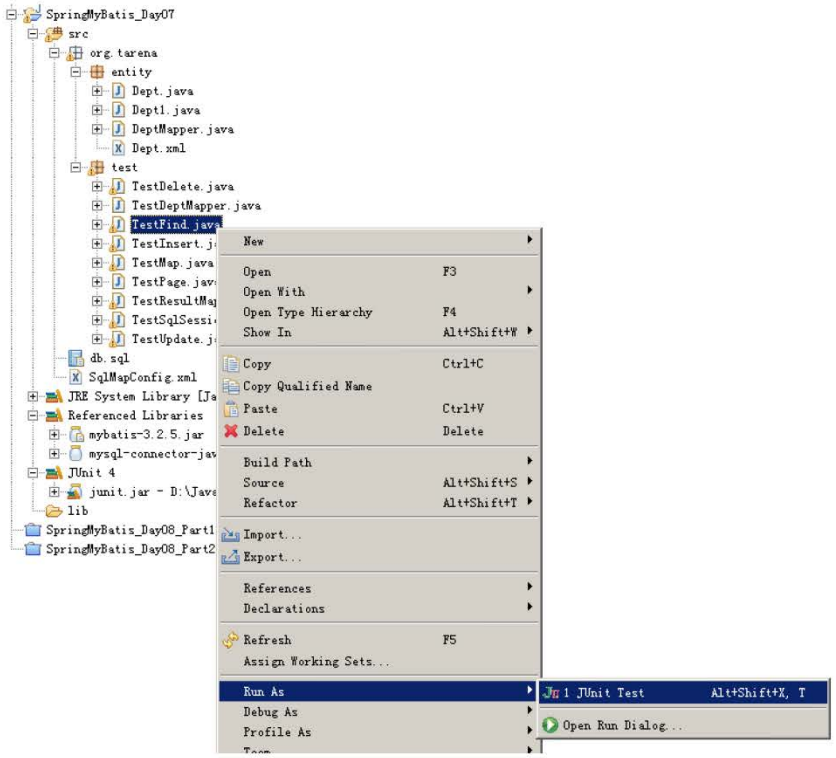


图-24

控制台输出如下信息，说明测试成功。如图-25 所示：

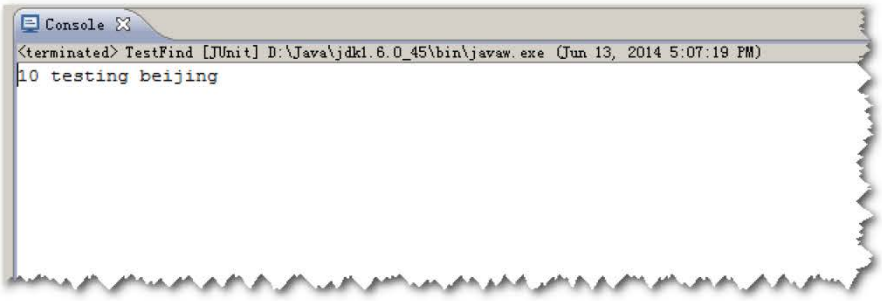


图-25

步骤十五：修改 DeptMapper 接口和 Dept.xml

DeptMapper 接口加入下图-26 方框内的代码：

```
package org.tarena.entity;

import java.util.List;

public interface DeptMapper {
    public void addDept(Dept dept);

    public void updateDept(Dept dept);

    public void deleteById(int id);

    public Dept findById(int id);

    public List<Dept> findAll();
}
```

图-26

DeptMapper 接口加入的代码如下：

```
public List<Dept> findAll();
```

Dept.xml 映射文件加入下图-27 方框内的配置：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">
    <insert id="addDept" parameterType="org.tarena.entity.Dept">
        insert into T_DEPT (DEPTNO,DNAME,LOC) values
        (#{deptno},#{dname},#{loc})
    </insert>

    <update id="updateDept" parameterType="org.tarena.entity.Dept">
        update T_DEPT set DNAME=#{dname},LOC=#{loc} where
        DEPTNO=#{deptno}
    </update>

    <delete id="deleteById" parameterType="int">
        delete from T_DEPT where DEPTNO=#{no}
    </delete>

    <select id="findById" parameterType="int"
        resultType="org.tarena.entity.Dept">
        select DEPTNO,DNAME,LOC from T_DEPT where DEPTNO=#{id}
    </select>

    <select id="findAll" resultType="org.tarena.entity.Dept">
        select DEPTNO,DNAME,LOC from T_DEPT
    </select>

</mapper>
```

图-27

Dept.xml 映射文件加入的代码如下：

```
<select id="findAll" resultType="org.tarena.entity.Dept">
    select DEPTNO,DNAME,LOC from T_DEPT
</select>
```

步骤十六：修改 TestFind 类

修改类 TestFind，加入图-28 方框内的代码：

```
package org.tarena.test;

import java.io.IOException;

public class TestFind {

    @Test
    public void testFindById() throws IOException {
        String conf = "SqlMapConfig.xml";
        Reader reader = Resources.getResourceAsReader(conf);
        // 创建SessionFactory对象
        SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        // 创建Session
        SqlSession session = sf.openSession();
        // 调用findById方法
        Dept dept = (Dept) session.selectOne("findById", 10);
        System.out.println(dept.getDeptno() + " " + dept.getDname() + " "
                + dept.getLoc());
        // 关闭
        session.close();
    }

    @Test
    public void testFindAll() throws IOException {
        String conf = "SqlMapConfig.xml";
        Reader reader = Resources.getResourceAsReader(conf);
        // 创建SessionFactory对象
        SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        // 创建Session
        SqlSession session = sf.openSession();
        // 调用findAll方法
        List<Dept> list = session.selectList("findAll");
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname() + " "
                    + dept.getLoc());
        }
        session.close();
    }
}
```

图-28

TestFind 类加入的代码如下：

```
@Test
public void testFindAll() throws IOException {
    String conf = "SqlMapConfig.xml";
    Reader reader = Resources.getResourceAsReader(conf);
    // 创建 SessionFactory 对象
    SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
    SqlSessionFactory sf = sfb.build(reader);
    // 创建 Session
    SqlSession session = sf.openSession();
    // 调用 findAll 方法
    List<Dept> list = session.selectList("findAll");
    for (Dept dept : list) {
        System.out.println(dept.getDeptno() + " " + dept.getDname() + " "
                + dept.getLoc());
    }
    session.close();
}
```

步骤十七：运行 TestFind 测试

在 TestFind 类右键运行测试，如图-29 所示：

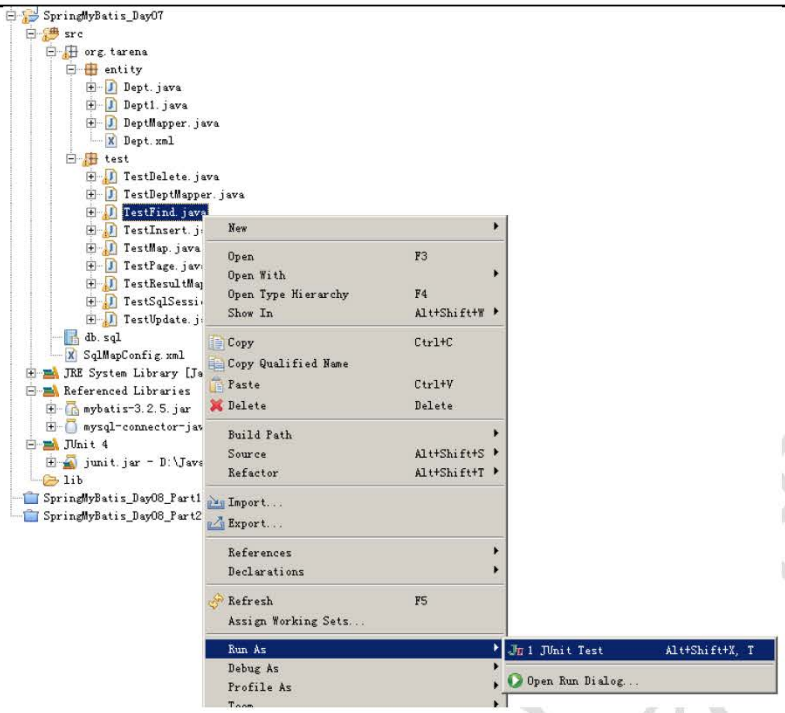


图-29

控制台输出方框内信息，说明测试成功。如图-30 所示：

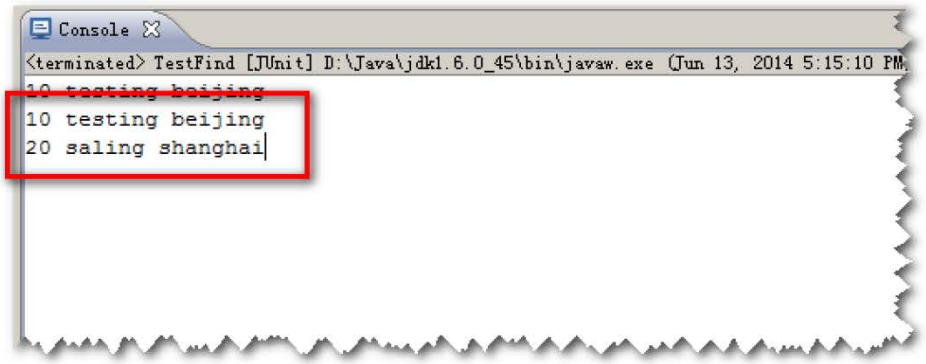


图-30

- **完整代码**

类 Dept 代码如下所示：

```
package org.tarena.entity;

public class Dept{
 private Integer deptno;
 private String dname;
 private String loc;
 public Integer getDeptno() {
     return deptno;
 }
 public void setDeptno(Integer deptno) {
     this.deptno = deptno;
 }
```

```
    public String getDname() {
        return dname;
    }
    public void setDname(String dname) {
        this.dname = dname;
    }
    public String getLoc() {
        return loc;
    }
    public void setLoc(String loc) {
        this.loc = loc;
    }
}
```

接口 DeptMapper 代码如下所示：

```
package org.tarena.entity;

import java.util.List;

public interface DeptMapper {
    public void addDept(Dept dept);

    public void updateDept(Dept dept);

    public void deleteById(int id);

    public Dept findById(int id);

    public List<Dept> findAll();

}
```

映射文件 Dept.xml 代码如下所示：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">
 <insert id="addDept" parameterType="org.tarena.entity.Dept">
     insert into T_DEPT (DEPTNO,DNAME,LOC) values
     (#{deptno},#{dname},#{loc})
 </insert>

 <update id="updateDept" parameterType="org.tarena.entity.Dept">
     update T_DEPT set DNAME=#{dname},LOC=#{loc} where
     DEPTNO=#{deptno}
 </update>

 <delete id="deleteById" parameterType="int">
     delete from T_DEPT where DEPTNO=#{no}
 </delete>

 <select id="findById" parameterType="int"
     resultType="org.tarena.entity.Dept">
     select DEPTNO,DNAME,LOC from T_DEPT where DEPTNO=#{id}
 </select>

 <select id="findAll" resultType="org.tarena.entity.Dept">
     select DEPTNO,DNAME,LOC from T_DEPT
 </select>

</mapper>
```

MyBatis 配置文件 SqlMapConfig.xml 代码如下所示：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration PUBLIC "-//ibatis.apache.org//DTD Config 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-config.dtd">
<configuration>
 <environments default="environment">
     <environment id="environment">
         <transactionManager type="JDBC" />
         <dataSource type="POOLED">
             <property name="driver" value="com.mysql.jdbc.Driver" />
             <property name="url"
                 value="jdbc:mysql://localhost:3306/test" />
             <property name="username" value="root" />
             <property name="password" value="root" />
         </dataSource>
     </environment>
 </environments>
 <mappers>
     <mapper resource="org/tarena/entity/Dept.xml" />
 </mappers>
</configuration>
```

测试类 TestInsert 代码如下所示：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.RowBounds;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestInsert {

 @Test
 public void testInsert() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader =
         Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb =
             new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     //调用 addDept 操作
     Dept dept = new Dept();
     dept.setDeptno(60);
     dept.setDname("testing");
     dept.setLoc("beijing");
     session.insert("addDept",dept);
     session.commit();
     //关闭
     session.close();
 }
}
```

测试类 TestUpdate 代码如下所示：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.RowBounds;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestUpdate {

 @Test
 public void testUpdate() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     //调用 findById 操作
     Dept dept = (Dept)session.selectOne("findById",60);
     dept.setDname("开发部");
     dept.setLoc("北京");
     //调用 updateDept 操作
     session.update("updateDept",dept);
     session.commit();
     //关闭
     session.close();
 }

}
```

测试类 TestDelete 代码如下所示：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.RowBounds;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestDelete {

 @Test
 public void testDelete() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
```

```
        SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        //创建 Session
        SqlSession session = sf.openSession();
        //调用 deleteById 操作
        session.delete("deleteById",60);
        session.commit();
        //关闭
        session.close();
    }
}
```

测试类 TestFind 代码如下所示：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestFind {

    @Test
    public void testFindById() throws IOException {
        String conf = "SqlMapConfig.xml";
        Reader reader = Resources.getResourceAsReader(conf);
        // 创建 SessionFactory 对象
        SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        // 创建 Session
        SqlSession session = sf.openSession();
        // 调用 findById 方法
        Dept dept = (Dept) session.selectOne("findById", 10);
        System.out.println(dept.getDeptno() + " " + dept.getDname() + " "
                + dept.getLoc());
        // 关闭
        session.close();
    }

    @Test
    public void testFindAll() throws IOException {
        String conf = "SqlMapConfig.xml";
        Reader reader = Resources.getResourceAsReader(conf);
        // 创建 SessionFactory 对象
        SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        // 创建 Session
        SqlSession session = sf.openSession();
        // 调用 findAll 方法
        List<Dept> list = session.selectList("findAll");
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname() + " "
                    + dept.getLoc());
        }
        session.close();
    }
}
```

3. 实现对 Dept 表的分页查询操作案例

- **问题**

当数据表记录很多时，需要将记录分页查询显示，那么在 MyBatis 中该如何实现呢？

- **方案**

MyBatis 提供了一个 RowBounds 类型，在查询时指定 RowBounds 参数对象可以控制抓取记录的起点和数量。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：新建 TestPage 类

新建类 TestPage，如图-31 所示：

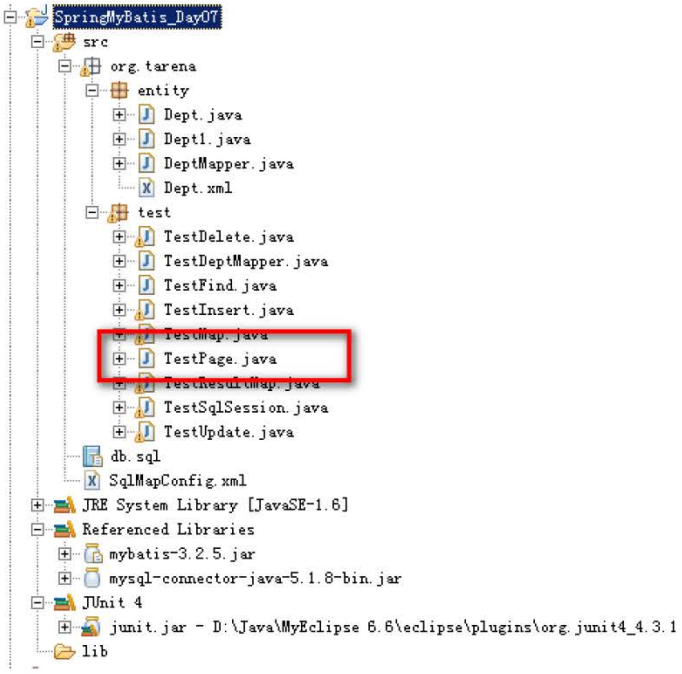


图-31

TestPage 类的代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.RowBounds;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
```

```
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestPage {

@Test
public void testFindPage() throws IOException{
    String conf = "SqlMapConfig.xml";
    Reader reader = Resources.getResourceAsReader(conf);
    //创建 SessionFactory 对象
    SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
    SqlSessionFactory sf = sfb.build(reader);
    //创建 Session
    SqlSession session = sf.openSession();

    int offset = 0;//起点,从 0 开始
    int limit = 2;//查几条
    RowBounds rowBounds = new RowBounds(offset, limit);
    List<Dept> list =
        session.selectList("findAll",null,rowBounds);
    for(Dept dept : list){
        System.out.println(dept.getDeptno()+" "
                +dept.getDname()+" "
                +dept.getLoc());
    }
    session.close();
}

}
```

步骤二：运行 TestPage 测试

在 TestPage 类右键运行测试，如图-32 所示：

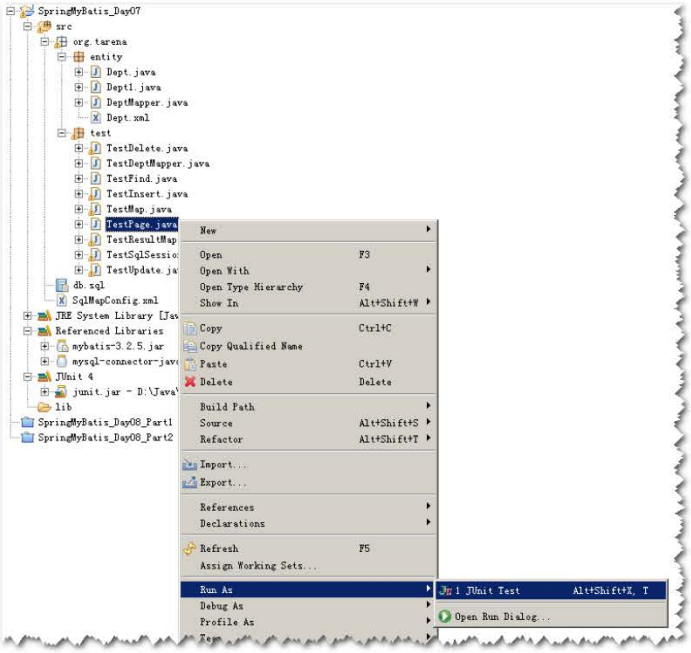


图-32

查看控制台，输出以下记录说明测试成功。如图-33 所示：

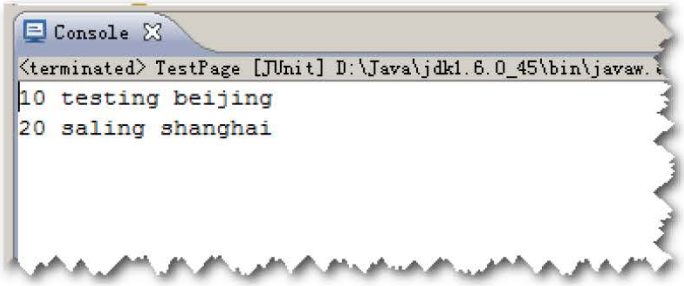


图-33

- **完整代码**

测试类 TestPage 代码如下所示：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.RowBounds;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;

public class TestPage {

 @Test
 public void testFindPage() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();

     int offset = 0;//起点,从 0 开始
     int limit = 2;//查几条
     RowBounds rowBounds = new RowBounds(offset, limit);
     List<Dept> list =
         session.selectList("findAll",null,rowBounds);
     for(Dept dept : list){
         System.out.println(dept.getDeptno()+" "
                 +dept.getDname()+" "
                 +dept.getLoc());
     }
     session.close();
 }

}
```

4. 实现对 Dept 表的 Map 类型查询操作案例

- **问题**

如何将查询结果以 Map 结构返回？

- **方案**

在 Mapper 映射文件中，定义 SQL 查询语句时，将 ResultType 返回类型属性指定成 java.util.HashMap。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：修改 Dept.xml

Dept.xml 映射文件加入下图-34 方框内的配置：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">
    <insert id="addDept" parameterType="org.tarena.entity.Dept">
        insert into T_DEPT (DEPTNO,DNAME,LOC) values
        (#{deptno},#{dname},#{loc})
    </insert>

    <update id="updateDept" parameterType="org.tarena.entity.Dept">
        update T_DEPT set DNAME=#{dname},LOC=#{loc} where
        DEPTNO=#{deptno}
    </update>

    <delete id="deleteById" parameterType="int">
        delete from T_DEPT where DEPTNO=#{no}
    </delete>

    <select id="findById" parameterType="int"
        resultType="org.tarena.entity.Dept">
        select DEPTNO,DNAME,LOC from T_DEPT where DEPTNO=#{id}
    </select>

    <select id="findAll" resultType="org.tarena.entity.Dept">
        select DEPTNO,DNAME,LOC from T_DEPT
    </select>

    <!-- 返回Map -->

    <select id="findDept" parameterType="int"
        resultType="java.util.HashMap">
        select DEPTNO,DNAME from T_DEPT where DEPTNO=#{no}
    </select>

</mapper>
```

图-34

Dept.xml 映射文件加入的代码如下：

```
<!-- 返回 Map -->
<select id="findDept" parameterType="int"
    resultType="java.util.HashMap">
    select DEPTNO,DNAME from T_DEPT where DEPTNO=#{no}
</select>
```

步骤二：新建 TestMap 类

新建类 TestMap，如图-35 所示：

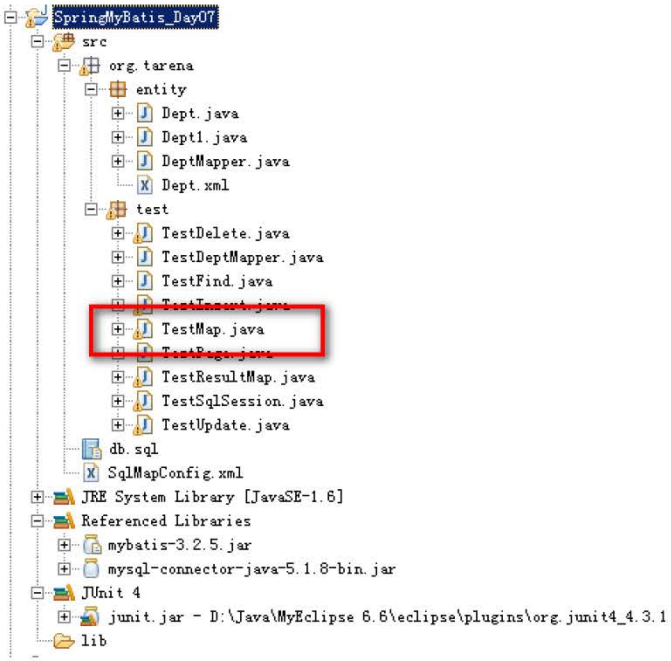


图-35

TestMap 类的代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;
import java.util.Map;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;

public class TestMap {

 @Test
 public void test2() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
```

```
        //创建 Session
        SqlSession session = sf.openSession();
        Map map = (Map)session.selectOne("findDept",10);
        System.out.println(map.get("DEPTNO")
                +" "+map.get("DNAME"));
        session.close();
    }
}
```

步骤三：运行 TestMap 测试

在 TestMap 类右键运行测试，如图-36 所示：

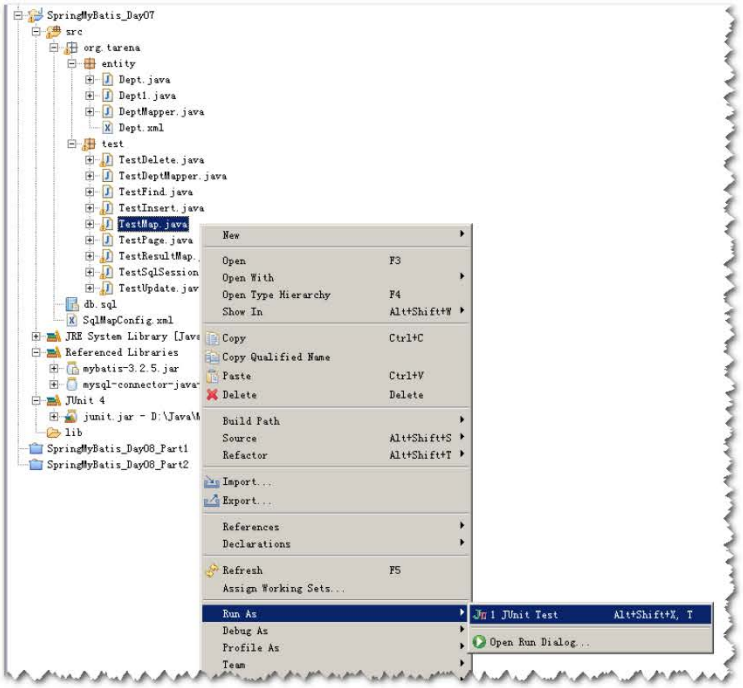


图-36

控制台输出如下信息，说明测试成功。如图-37 所示：

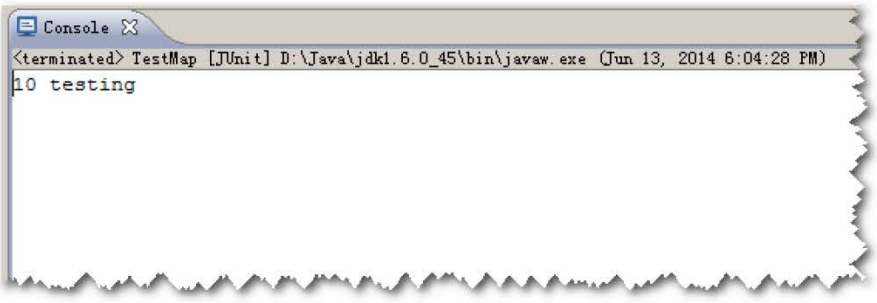


图-37

步骤四：修改 Dept.xml

Dept.xml 映射文件加入下图-38 方框内的配置：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">
    <insert id="addDept" parameterType="org.tarena.entity.Dept">
        insert into T_DEPT (DEPTNO,DNAME,LOC) values
        (#{deptno},#{dname},#{loc})
    </insert>

    <update id="updateDept" parameterType="org.tarena.entity.Dept">
        update T_DEPT set DNAME=#{dname},LOC=#{loc} where
        DEPTNO=#{deptno}
    </update>

    <delete id="deleteById" parameterType="int">
        delete from T_DEPT where DEPTNO=#{no}
    </delete>

    <select id="findById" parameterType="int"
        resultType="org.tarena.entity.Dept">
        select DEPTNO,DNAME,LOC from T_DEPT where DEPTNO=#{id}
    </select>

    <select id="findAll" resultType="org.tarena.entity.Dept">
        select DEPTNO,DNAME,LOC from T_DEPT
    </select>

    <!-- 返回Map -->

    <select id="findDept" parameterType="int"
        resultType="java.util.HashMap">
        select DEPTNO,DNAME from T_DEPT where DEPTNO=#{no}
    </select>

    <select id="findDepts" resultType="java.util.HashMap">
        select DEPTNO,DNAME from T_DEPT
    </select>
</mapper>
```

图-38

Dept.xml 映射文件加入的代码如下：

```
<select id="findDepts" resultType="java.util.HashMap">
    select DEPTNO,DNAME from T_DEPT
</select>
```

步骤五：修改 TestMap 类

修改类 TestMap，加入图-39 方框内的代码：

```
package org.tarena.test;

import java.io.IOException;

public class TestMap {

    @Test
    public void test2() throws IOException{
        String conf = "SqlMapConfig.xml";
        Reader reader =
            Resources.getResourceAsReader(conf);
        //创建SessionFactory对象
        SqlSessionFactoryBuilder sfb =
                new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        //创建Session
        SqlSession session = sf.openSession();
        Map map = (Map)session.selectOne("findDept",10);
        System.out.println(map.get("DEPTNO")
                +" "+map.get("DNAME"));
        session.close();
    }

    @Test
    public void test1() throws IOException{
        String conf = "SqlMapConfig.xml";
        Reader reader =
            Resources.getResourceAsReader(conf);
        //创建SessionFactory对象
        SqlSessionFactoryBuilder sfb =
                new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        //创建Session
        SqlSession session = sf.openSession();
        List<Map> list = session.selectList("findDepts");
        for(Map map : list){
            System.out.println(map.get("DEPTNO")
                +" "+map.get("DNAME"));
        }
        session.close();
    }
}
```

图-39

TestMap 类加入的代码如下：

```
@Test
public void test1() throws IOException{
    String conf = "SqlMapConfig.xml";
    Reader reader = Resources.getResourceAsReader(conf);
    //创建 SessionFactory 对象
    SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
    SqlSessionFactory sf = sfb.build(reader);
    //创建 Session
    SqlSession session = sf.openSession();
    List<Map> list = session.selectList("findDepts");
    for(Map map : list){
        System.out.println(map.get("DEPTNO")
            +" "+map.get("DNAME"));
    }
    session.close();
}
```

步骤六：运行 TestMap 测试

在 TestMap 类右键运行测试，如图-40 所示：

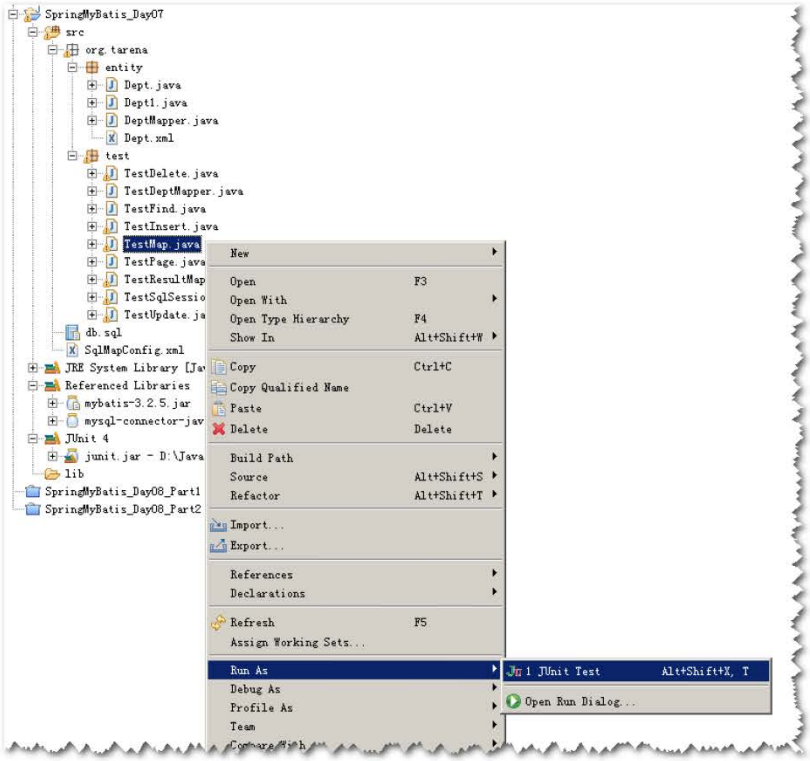


图-40

控制台输出方框内信息，说明测试成功。如图-41 所示：

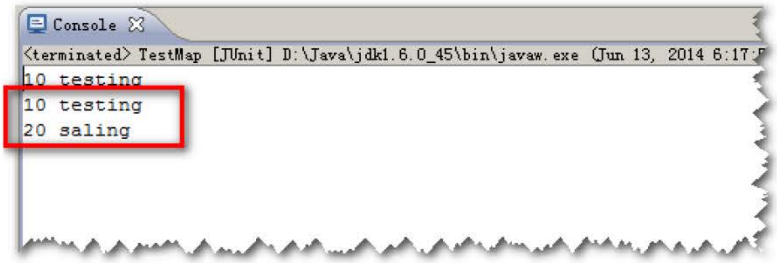


图-41

- **完整代码**

映射文件 Dept.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">

 <insert id="addDept" parameterType="org.tarena.entity.Dept">
     insert into T_DEPT (DEPTNO,DNAME,LOC) values
     (#{deptno},#{dname},#{loc})
 </insert>
```

```
<update id="updateDept" parameterType="org.tarena.entity.Dept">
    update T DEPT set DNAME=#{dname},LOC=#{loc} where
    DEPTNO=#{deptno}
</update>

<delete id="deleteById" parameterType="int">
    delete from T DEPT where DEPTNO=#{no}
</delete>

<select id="findById" parameterType="int"
    resultType="org.tarena.entity.Dept">
    select DEPTNO,DNAME,LOC from T DEPT where DEPTNO=#{id}
</select>

<select id="findAll" resultType="org.tarena.entity.Dept">
    select DEPTNO,DNAME,LOC from T_DEPT
</select>

<!-- 返回 Map -->
<select id="findDept" parameterType="int"
    resultType="java.util.HashMap">
    select DEPTNO,DNAME from T DEPT where DEPTNO=#{no}
</select>

<select id="findDepts" resultType="java.util.HashMap">
    select DEPTNO,DNAME from T DEPT
</select>

</mapper>
```

测试类 TestMap 代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;
import java.util.Map;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;

public class TestMap {
 @Test
 public void test2() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     Map map = (Map)session.selectOne("findDept",10);
     System.out.println(map.get("DEPTNO")
             +" "+map.get("DNAME"));
     session.close();
 }

 @Test
 public void test1() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
```

```
        //创建 SessionFactory 对象
        SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        //创建 Session
        SqlSession session = sf.openSession();
        List<Map> list = session.selectList("findDepts");
        for(Map map : list){
            System.out.println(map.get("DEPTNO")
                +" "+map.get("DNAME"));
        }
        session.close();
    }
}
```

5. 使用 Mapper 对 Dept 表操作案例

- **问题**

如何使用 Mapper 接口对数据表操作。

- **方案**

首先根据 Mapper 映射描述文件编写一个 Mapper 接口，接口方法名和 SQL 映射描述定义的 id 属性保持一致。

然后利用 SqlSession 提供的 getMapper(..)方法，会自动返回一个 Mapper 接口实例。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：新建 TestDeptMapper 类

新建类 TestDeptMapper，如图-42 所示：

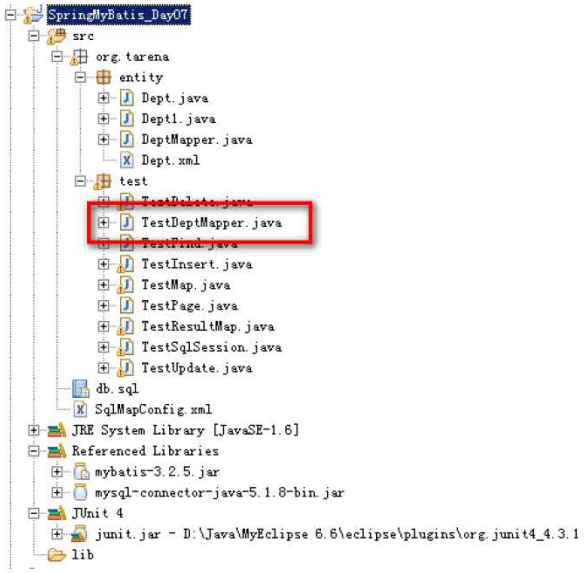


图-42

TestDeptMapper 类的代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;
import org.tarena.entity.DeptMapper;

public class TestDeptMapper {
 @Test
 public void testFindAll() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     DeptMapper mapper = session.getMapper(DeptMapper.class);
     //调用 findAll 方法
     List<Dept> list = mapper.findAll();
     for(Dept dept : list){
         System.out.println(dept.getDeptno()+" "
                 +dept.getDname()+" "
                 +dept.getLoc());
     }
     session.close();
 }
}
```

步骤二：运行 TestDeptMapper 测试

在 TestDeptMapper 类右键运行测试，如图-43 所示：

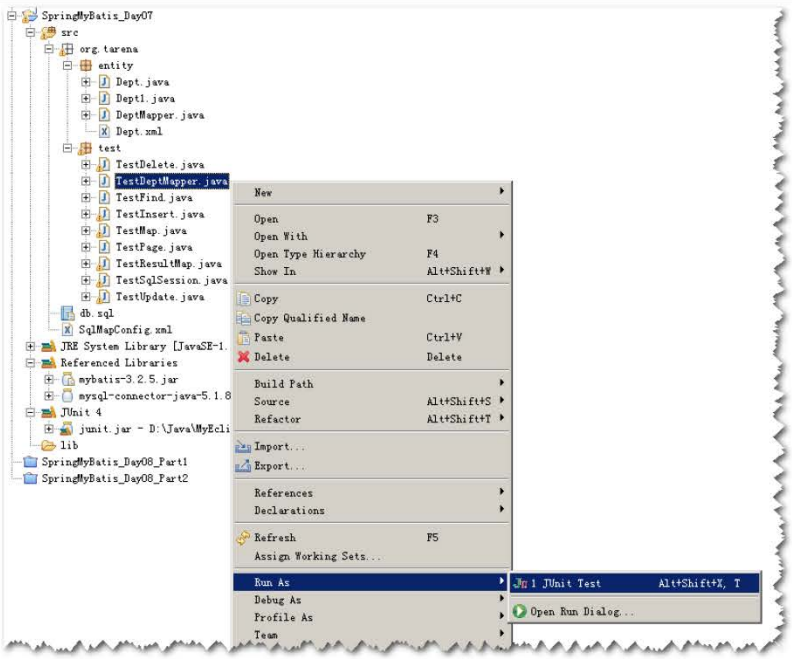


图-43

查看控制台，输出以下记录说明测试成功。如图-44 所示：

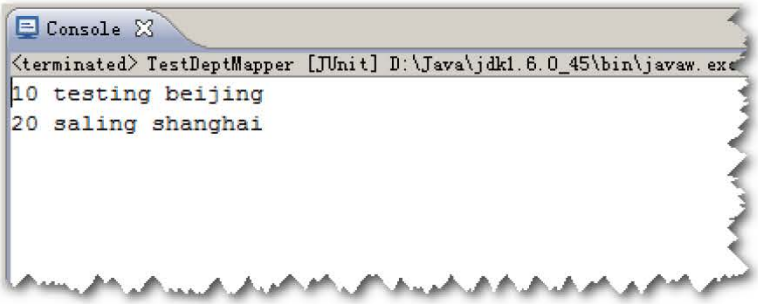


图-44

步骤三：修改 TestDeptMapper 测试类

TestDeptMapper 测试类文件加入下图-45 方框内的配置：

```
package org.tarena.test;

import java.io.IOException;

public class TestDeptMapper {

    public void testFindAll() throws IOException{

    @Test
    public void testInsert() throws IOException{
        String conf = "SqlMapConfig.xml";
        Reader reader =
            Resources.getResourceAsReader(conf);
        //创建SessionFactory对象
        SqlSessionFactoryBuilder sfb =
                new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        //创建Session
        SqlSession session = sf.openSession();
        DeptMapper mapper =
            session.getMapper(DeptMapper.class);
        //调用findById方法
        Dept dept = new Dept();
        dept.setDeptno(60);
        dept.setDname("testing");
        dept.setLoc("beijing");
        mapper.addDept(dept);
        session.commit();
        //关闭
        session.close();
    }
}
```

图-45

TestDeptMapper 测试类添加代码如下：

```
@Test
public void testInsert() throws IOException{
    String conf = "SqlMapConfig.xml";
    Reader reader = Resources.getResourceAsReader(conf);
    //创建 SessionFactory 对象
    SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
    SqlSessionFactory sf = sfb.build(reader);
    //创建 Session
    SqlSession session = sf.openSession();
    DeptMapper mapper = session.getMapper(DeptMapper.class);
    //调用 findById 方法
    Dept dept = new Dept();
    dept.setDeptno(60);
    dept.setDname("testing");
    dept.setLoc("beijing");
    mapper.addDept(dept);
    session.commit();
    //关闭
    session.close();
}
```

步骤四：运行 TestDeptMapper 测试

在 TestMap 类右键运行测试，如图-46 所示：

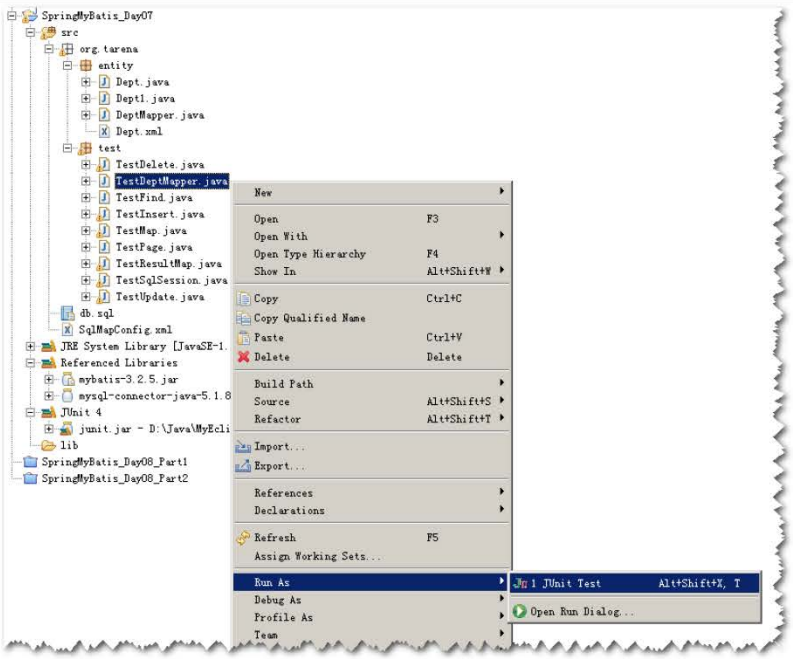


图-46

查看数据库，增加图-47 方框内的记录说明测试成功。

deptno	dname	loc
10	testing	beijing
60	testing	beijing

图-47

修改 TestDeptMapper 测试类：

TestDeptMapper 测试类文件加入下图-48 方框内的配置：

```
package org.tarena.test;

import java.io.IOException;

public class TestDeptMapper {

    public void testFindAll() throws IOException{


    public void testInsert() throws IOException{

    @Test
    public void testDelete() throws IOException{
        String conf = "SqlMapConfig.xml";
        Reader reader =
            Resources.getResourceAsReader(conf);
        //创建SessionFactory对象
        SqlSessionFactoryBuilder sfb =
                new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        //创建Session
        SqlSession session = sf.openSession();
        DeptMapper mapper =
            session.getMapper(DeptMapper.class);
        //调用deleteById方法
        mapper.deleteById(60);
        session.commit();
        //关闭
        session.close();
    }

}
```

图-48

TestDeptMapper 测试类添加代码如下：

```
@Test
public void testDelete() throws IOException{
    String conf = "SqlMapConfig.xml";
    Reader reader =
        Resources.getResourceAsReader(conf);
    //创建 SessionFactory 对象
    SqlSessionFactoryBuilder sfb =
            new SqlSessionFactoryBuilder();
    SqlSessionFactory sf = sfb.build(reader);
    //创建 Session
    SqlSession session = sf.openSession();
    DeptMapper mapper =
        session.getMapper(DeptMapper.class);
    //调用 deleteById 方法
    mapper.deleteById(60);
    session.commit();
    //关闭
    session.close();
}
```

步骤五：运行 TestDeptMapper 测试

在 TestMap 类右键运行测试，如图-49 所示：

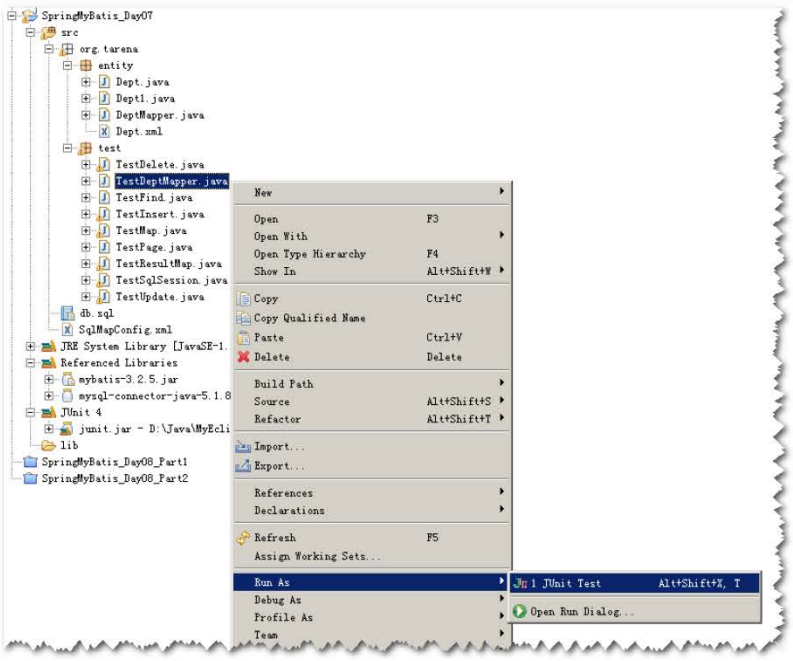


图-49

查看数据库，deptno 为 60 的记录被删除，说明测试成功。

- **完整代码**

测试类 TestDeptMapper 代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;
import org.tarena.entity.DeptMapper;

public class TestDeptMapper {

 @Test
 public void testFindAll() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     DeptMapper mapper = session.getMapper(DeptMapper.class);
     //调用 findAll 方法
     List<Dept> list = mapper.findAll();
     for(Dept dept : list){
         System.out.println(dept.getDeptno()+" "
```

```
                +dept.getDname()+" "
                +dept.getLoc());
        }
        session.close();
    }


    @Test
    public void testInsert() throws IOException{
        String conf = "SqlMapConfig.xml";
        Reader reader = Resources.getResourceAsReader(conf);
        //创建SessionFactory对象
        SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        //创建Session
        SqlSession session = sf.openSession();
        DeptMapper mapper = session.getMapper(DeptMapper.class);
        //调用findById方法
        Dept dept = new Dept();
        dept.setDeptno(60);
        dept.setDname("testing");
        dept.setLoc("beijing");
        mapper.addDept(dept);
        session.commit();
        //关闭
        session.close();
    }

    @Test
    public void testDelete() throws IOException{
        String conf = "SqlMapConfig.xml";
        Reader reader = Resources.getResourceAsReader(conf);
        //创建SessionFactory对象
        SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
        SqlSessionFactory sf = sfb.build(reader);
        //创建Session
        SqlSession session = sf.openSession();
        DeptMapper mapper = session.getMapper(DeptMapper.class);
        //调用deleteById方法
        mapper.deleteById(60);
        session.commit();
        //关闭
        session.close();
    }
}
```

6. 利用 ResultMap 自定义映射案例

- **问题**

当实体类属性和数据表字段名不一致时，该如何将查询结果映射成 Bean 对象。

- **方案**

在 Mapper 映射描述文件中，利用<resultMap>元素可以指定字段值和属性之间的对应关系。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：新建 Dept1 实体类

新建实体类 Dept1，如图-50 所示：：

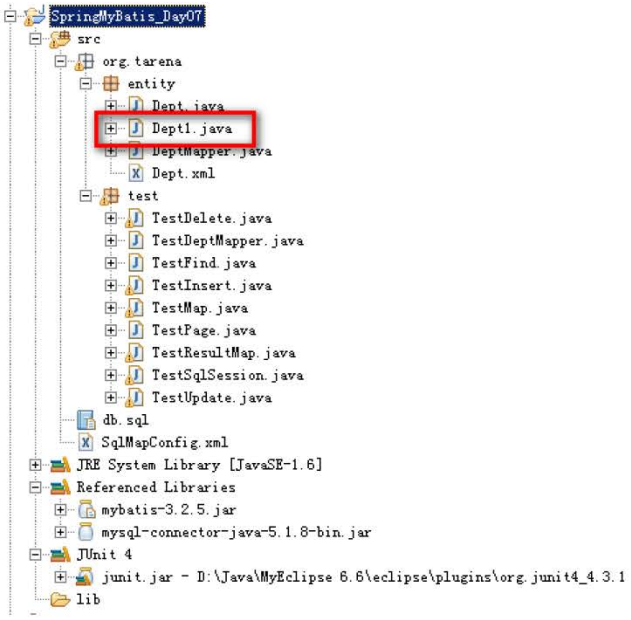


图-50

Dept1 实体类的代码如下所示：

```
package org.tarena.entity;

public class Dept1 {
 private Integer no;
 private String name;
 private String loc;

 public Integer getNo() {
     return no;
 }
 public void setNo(Integer no) {
     this.no = no;
 }
 public String getName() {
     return name;
 }
 public void setName(String name) {
     this.name = name;
 }
 public String getLoc() {
     return loc;
 }
 public void setLoc(String loc) {
     this.loc = loc;
 }
}
```

步骤二：修改 Dept.xml

Dept.xml 映射文件加入下图-51 的配置：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">

    <insert id="addDept" parameterType="org.tarena.entity.Dept">
    </insert>

    <update id="updateDept" parameterType="org.tarena.entity.Dept">
    </update>

    <delete id="deleteById" parameterType="int">
    </delete>

    <select id="findById" parameterType="int"
    </select>

    <select id="findAll" resultType="org.tarena.entity.Dept">
    </select>

    <!-- 返回Map -->

    <select id="findDept" parameterType="int"
    </select>

    <select id="findDepts" resultType="java.util.HashMap">
    </select>

    <!-- ResultMap应用 -->
    <resultMap id="deptMap" type="org.tarena.entity.Dept1">
        <result property="no" column="DEPTNO" jdbcType="INTEGER"
            javaType="int" />
        <result property="name" column="DNAME" jdbcType="VARCHAR"
            javaType="string" />
        <result property="loc" column="LOC" jdbcType="VARCHAR"
            javaType="string" />
    </resultMap>
    <select id="findAll1" resultMap="deptMap">
        select DEPTNO,DNAME,LOC from T_DEPT
    </select>

</mapper>
```

图-51

Dept.xml 映射文件加入的代码如下：

```
<!-- ResultMap 应用 -->
<resultMap id="deptMap" type="org.tarena.entity.Dept1">
    <result property="no" column="DEPTNO" jdbcType="INTEGER"
        javaType="int" />
    <result property="name" column="DNAME" jdbcType="VARCHAR"
        javaType="string" />
    <result property="loc" column="LOC" jdbcType="VARCHAR"
        javaType="string" />
</resultMap>
<select id="findAll1" resultMap="deptMap">
    select DEPTNO,DNAME,LOC from T_DEPT
</select>
```

步骤三：新建 TestResultMap 类

新建类 TestResultMap，如图-52 所示：

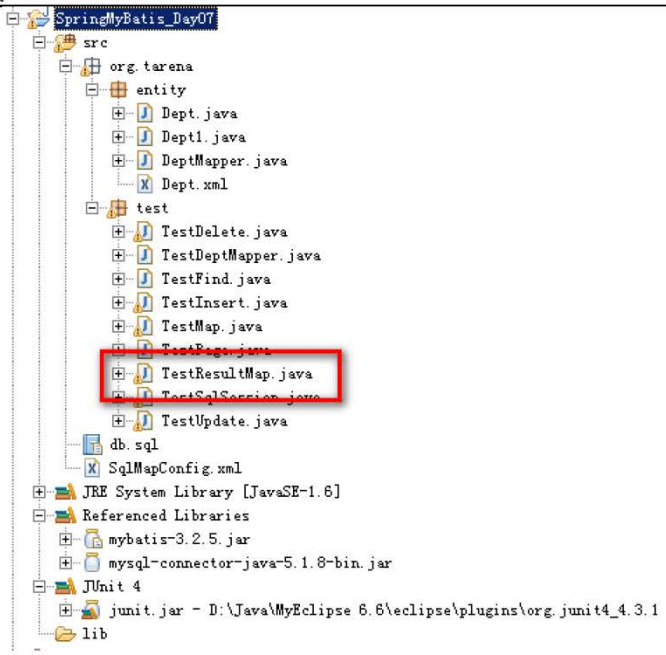


图-52

TestResultMap 类的代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;
import org.tarena.entity.Dept1;
import org.tarena.entity.DeptMapper;

public class TestResultMap {
 @Test
 public void testFindAll() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     DeptMapper mapper = session.getMapper(DeptMapper.class);
     //调用 findAll 方法
     List<Dept1> list = mapper.findAll1();
     for(Dept1 dept : list){
         System.out.println(dept.getNo()+" "
                 +dept.getName()+" "
                 +dept.getLoc());
     }
     session.close();
 }
}
```

步骤四：运行 TestResultMap 测试

在 TestResultMap 类右键运行测试，如图-53 所示：

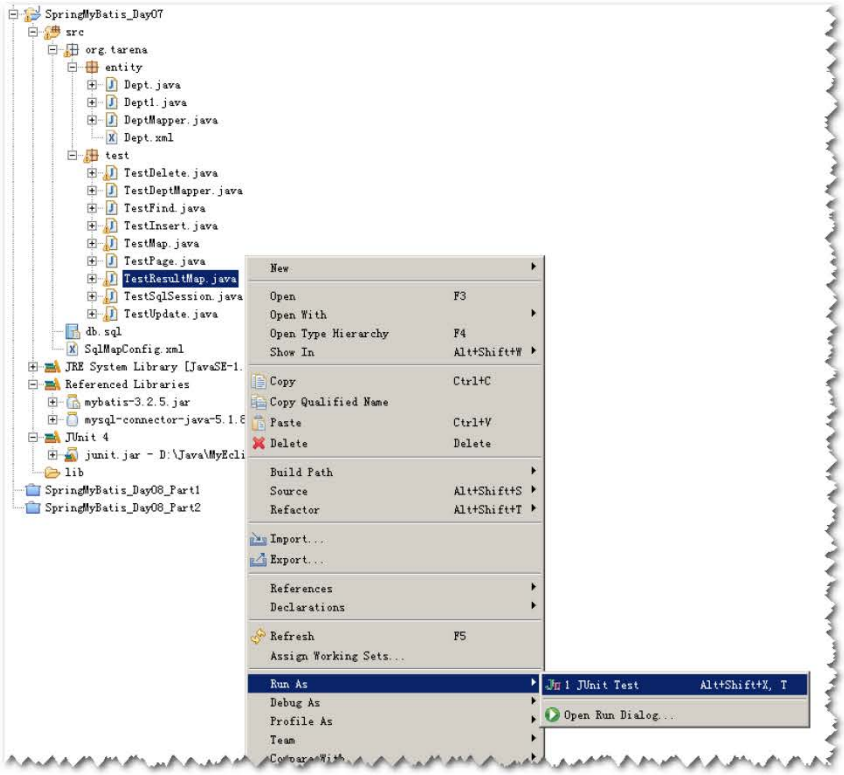


图-53

控制台输出如下信息，说明测试成功。如图-54 所示：

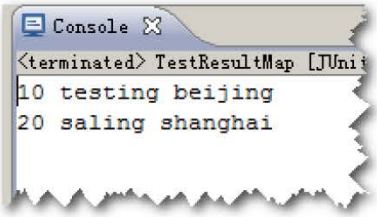


图-54

- **完整代码**

实体类 Dept1 代码如下：

```
package org.tarena.entity;

public class Dept1 {
 private Integer no;
 private String name;
 private String loc;

 public Integer getNo() {
     return no;
```

```
    }
    public void setNo(Integer no) {
        this.no = no;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getLoc() {
        return loc;
    }
    public void setLoc(String loc) {
        this.loc = loc;
    }
}
```

映射文件 Dept.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.entity.DeptMapper">

<insert id="addDept" parameterType="org.tarena.entity.Dept">
    insert into T_DEPT (DEPTNO,DNAME,LOC) values
    (#{deptno},#{dname},#{loc})
</insert>

<update id="updateDept" parameterType="org.tarena.entity.Dept">
    update T_DEPT set DNAME=#{dname},LOC=#{loc} where
    DEPTNO=#{deptno}
</update>

<delete id="deleteById" parameterType="int">
    delete from T DEPT where DEPTNO=#{no}
</delete>

<select id="findById" parameterType="int"
    resultType="org.tarena.entity.Dept">
    select DEPTNO,DNAME,LOC from T_DEPT where DEPTNO=#{id}
</select>

<select id="findAll" resultType="org.tarena.entity.Dept">
    select DEPTNO,DNAME,LOC from T_DEPT
</select>

<!-- 返回 Map -->
<select id="findDept" parameterType="int"
    resultType="java.util.HashMap">
    select DEPTNO,DNAME from T_DEPT where DEPTNO=#{no}
</select>

<select id="findDepts" resultType="java.util.HashMap">
    select DEPTNO,DNAME from T DEPT
</select>

<!-- ResultMap 应用 -->
<resultMap id="deptMap" type="org.tarena.entity.Dept1">
    <result property="no" column="DEPTNO" jdbcType="INTEGER"
        javaType="int" />
    <result property="name" column="DNAME" jdbcType="VARCHAR"
        javaType="string" />
    <result property="loc" column="LOC" jdbcType="VARCHAR"
        javaType="string" />
```

```
</resultMap>
<select id="findAll1" resultMap="deptMap">
    select DEPTNO,DNAME,LOC from T_DEPT
</select>

</mapper>
```

测试类 TestResultMap 代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.io.Reader;
import java.util.List;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Test;
import org.tarena.entity.Dept;
import org.tarena.entity.Dept1;
import org.tarena.entity.DeptMapper;

public class TestResultMap {
 @Test
 public void testFindAll() throws IOException{
     String conf = "SqlMapConfig.xml";
     Reader reader = Resources.getResourceAsReader(conf);
     //创建 SessionFactory 对象
     SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
     SqlSessionFactory sf = sfb.build(reader);
     //创建 Session
     SqlSession session = sf.openSession();
     DeptMapper mapper = session.getMapper(DeptMapper.class);
     //调用 findAll 方法
     List<Dept1> list = mapper.findAll1();
     for(Dept1 dept : list){
         System.out.println(dept.getNo()+" "
                 +dept.getName()+" "
                 +dept.getLoc());
     }
     session.close();
 }
}
```

课后作业

1. 列举 MyBatis 的常用 API 及方法。

2. 利用 MyBatis 实现对员工表 T_EMP 的 CRUD 基本操作。

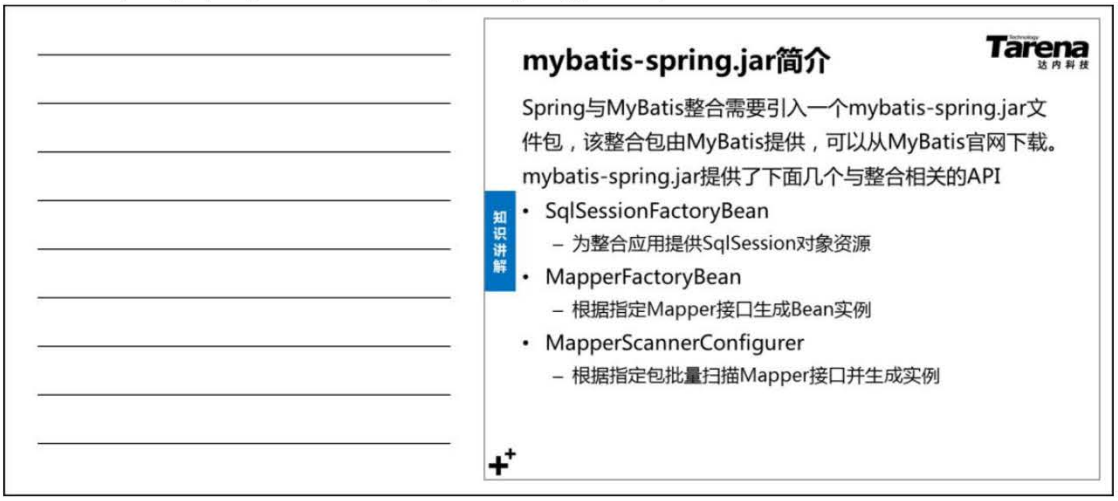
MyBatis 核心

Unit03

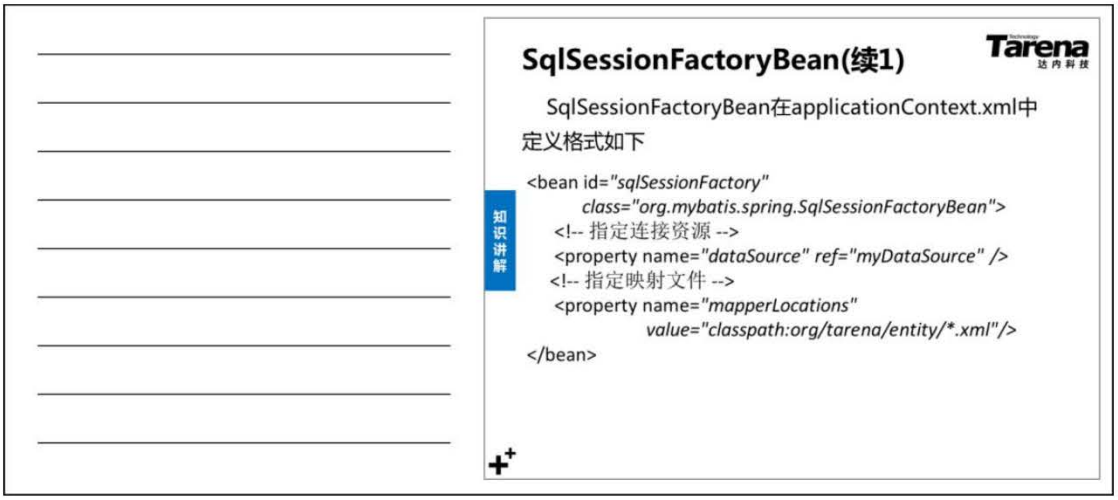
1. Spring + MyBatis

1.1. Spring 与 MyBatis 整合

1.1.1. 【Spring 与 MyBatis 整合】mybatis-spring.jar 简介



1.1.2. 【Spring 与 MyBatis 整合】SqlSessionFactoryBean



SqlSessionFactoryBean（续2）

Tarena 达内科技

通过SqlSessionFactoryBean还可以定义一些属性来指定Mybatis框架的配置信息。在定义SqlSessionFactoryBean时，可以使用以下常用属性

知识讲解

属性名	描述
dataSource	用于指定连接数据库的数据源(必要属性)
mapperLocations	用于指定Mapper文件存放的位置
configLocation	用于指定Mybatis的配置文件位置。如果指定了该属性，那么会以该配置文件的内容作为配置信息构建对应的SqlSessionFactoryBuilder，但是后续属性指定的内容会覆盖该配置文件里面指定的对应内容
typeAliasesPackage	它一般对应我们的实体类所在的包，这个时候会自动取对应包中不包括包名的简单类名作为包括包名的别名。多个package之间可以用逗号或者分号等来进行分隔
typeAliases	数组类型，用来指定别名的。指定了这个属性后，Mybatis会把这个类型的短名称作为这个类型的别名

1.1.3. 【Spring 与 MyBatis 整合】MapperFactoryBean

MapperFactoryBean

Tarena 达内科技

- MapperFactoryBean作用是根据Mapper接口获取我们想要的Mapper对象，它封装了原有session.getMapper()功能实现。

知识讲解

- 在定义MapperFactoryBean时，需要注入以下两个属性
  - 一个是SqlSessionFactoryBean对象，用于提供SqlSession
  - 一个是要返回Mapper对象的Mapper接口

MapperFactoryBean（续1）

Tarena 达内科技

MapperFactoryBean在applicationContext.xml中定义格式如下

知识讲解

```
<bean id="deptMapper"
      class="org.mybatis.spring.mapper.MapperFactoryBean">
  <!-- 指定Mapper接口 -->
  <property name="mapperInterface"
            value="org.tarena.mapper.DeptMapper" />
  <!-- 指定SqlSessionFactoryBean对象 -->
  <property name="sqlSessionFactory"
            ref="sqlSessionFactory" />
</bean>
```

1.1.4. 【Spring 与 MyBatis 整合】MapperScannerConfigurer

MapperScannerConfigurer

解说知

- 在使用MapperFactoryBean时，有一个Mapper就需要定义一个对应的MapperFactoryBean。当Mapper比较少时可以，但遇到大量Mapper时就需要使用mybatis-spring.jar提供的MapperScannerConfigurer组件，通过这个组件会自动扫描各个Mapper接口，并注册对应的MapperFactoryBean对象

MapperScannerConfigurer（续1）

解说知

- 在定义MapperScannerConfigurer时，只需要指定一个basePackage即可。basePackage用于指定Mapper接口所在的包，在这个包及其所有子包下面的Mapper接口都将被搜索到，并把它们注册为一个个MapperFactoryBean对象。多个包之间可以使用逗号或者分号进行分隔。

```
<bean
  class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="org.tarena.mapper" />
</bean>
```

- 提示：sqlSessionFactory属性可以不用指定，会以Autowired方式自动注入

MapperScannerConfigurer（续2）

解说知

如果指定的某个包下并不完全是我们定义的Mapper接口，此时使用MapperScannerConfigurer的另外两个属性可以缩小搜索和注册范围，一个是annotationClass，另一个是markerInterface。

- annotationClass：用于指定一个注解标记，当指定了annotationClass时，MapperScannerConfigurer将只注册使用了annotationClass注解标记的接口
- markerInterface：用于指定一个接口，当指定了markerInterface时，MapperScannerConfigurer将只注册继承自markerInterface的接口

MapperScannerConfigurer（续3）

MapperScannerConfigurer定义示例

```
<bean
  class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="org.tarena" />
    <property name="annotationClass"
              value="org.tarena.annotation.MyBatisRepository"/>
</bean>
```

解说知识

上面配置含义：MapperScannerConfigurer自动扫描org.tarena包下所有接口，遇到带@MyBatisRepository标记的将对应MapperFactoryBean对象注册

MapperScannerConfigurer（续4）

@MyBatisRepository自定义注解标记代码如下

```
public @interface MyBatisRepository {

}
```

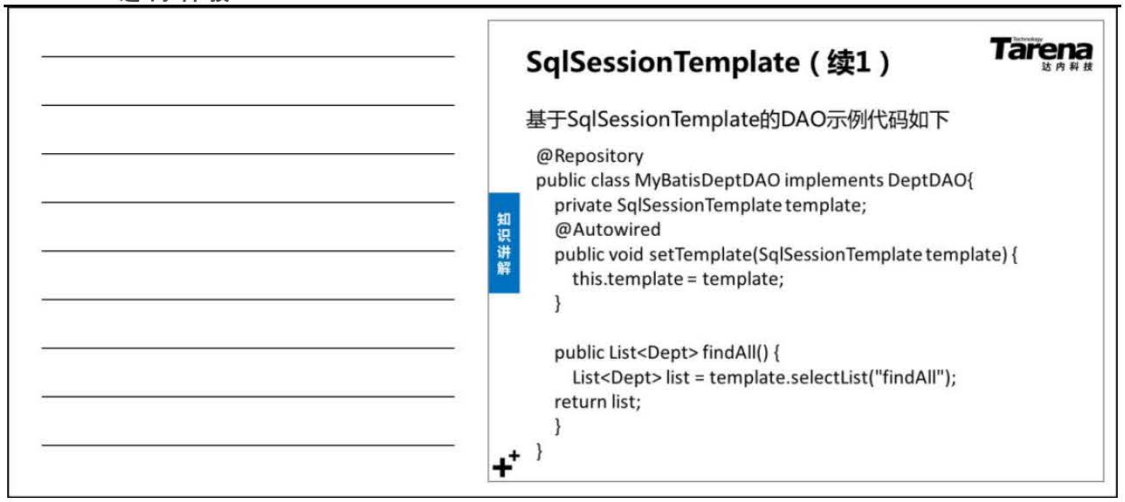
解说知识

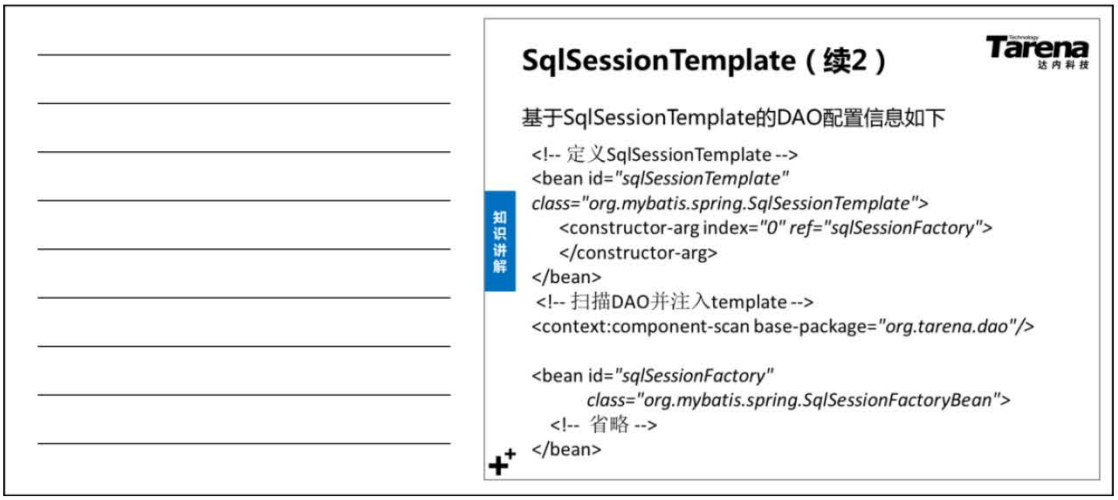
1.1.5. 【Spring 与 MyBatis 整合】SqlSessionTemplate

SqlSessionTemplate

解说知识

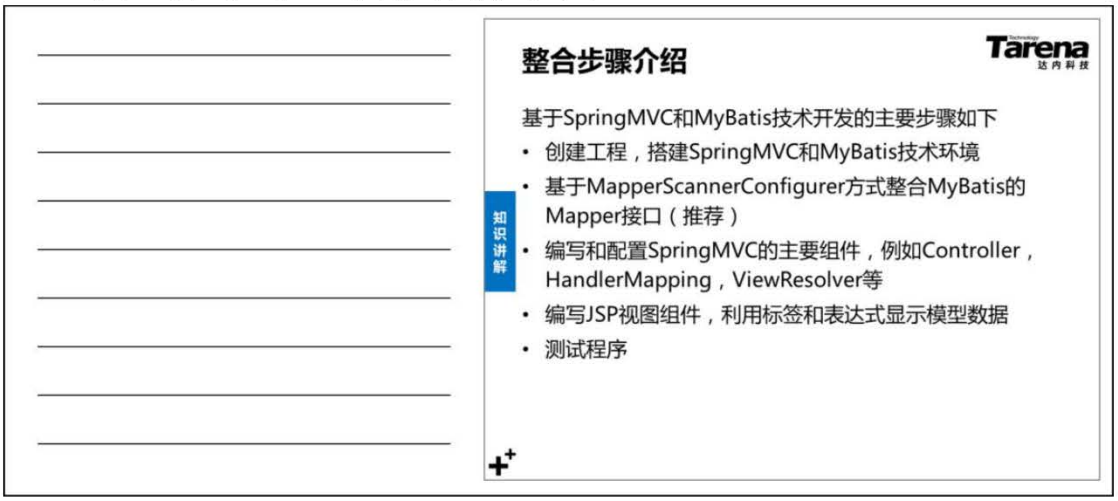
上述整合完成后，程序可直接使用Spring容器中的Mapper接口实例进行编程。此外mybatis-spring.jar还提供了一个SqlSessionTemplate组件，也可以将该组件对象注入给程序中的DAO，在DAO中利用SqlSessionTemplate编程。





1.2. Spring 整合 MyBatis 应用

1.2.1. 【Spring 整合 MyBatis 应用】整合步骤介绍



1.2.2. 【Spring 整合 MyBatis 应用】整合步骤介绍

整合步骤介绍（续1）

如何搭建SpringMVC和MyBatis技术环境？

- 创建一个Web工程
- 添加MyBatis相关技术环境
  - 引入数据库驱动包和MyBatis开发包
  - 引入dbcp连接池开发包
- 添加SpringMVC相关技术环境
  - 引入Spring ioc,jdbc,tx等支持的开发包
  - 引入Spring webmvc支持的开发包
  - 在src下添加applicationContext.xml配置文件
  - 在web.xml中配置DispatcherServlet主控制器
- 引入MyBatis和Spring整合开发包mybatis-spring.jar

整合步骤介绍（续2）

如何基于MapperScannerConfigurer方式整合MyBatis的Mapper接口？

- 根据数据表编写实体类
- 编写Mapper映射文件，在XML中添加SQL操作的定义
- 编写Mapper接口，定义SQL操作方法
- 在Spring配置文件中定义以下Bean
  - DataSource
  - SqlSessionFactoryBean
  - MapperScannerConfigurer
- 测试Spring容器的DAO组件

整合步骤介绍（续3）

如何编写和配置SpringMVC的主要组件？

- 编写Controller和请求处理方法
- 配置<mvc:annotation-driven/>，支持@RequestMapping
- 配置Controller组件
  - 开启组件扫描，将Controller扫描到Spring容器
  - 需要DAO时采用注入方式使用
  - 在请求处理方法上使用@RequestMapping指定对应的请求
- 配置ViewResolver组件

整合步骤介绍（续4）

Tarena 达内科技

在JSP视图组件中如何显示模型数据？

- JSP可以使用JSTL标签库，需要引入开发包
- JSP可以使用EL表达式
- JSP可以使用SpringMVC的表单标签

知识讲解

经典案例

1. 通过 spring 的 SqlSessionFactoryBean 和 MapperFactoryBean 实现对 Dept 表查询操作案例

- **问题**

如何将 Spring 和 MyBatis 整合在一起应用？

- **方案**

利用 SqlSessionFactoryBean 和 MapperFactoryBean 组件可以完成 Spring 与 MyBatis 的整合。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：确认在 MySQL 的 test 数据库中有之前创建的 t_dept 表和数据

如果 MySQL 的 test 数据库中没有 t_dept 表执行以下 SQL 语句，用来创建表和相关数据：

```
create table t_dept(
  deptno int primary key,
  dname varchar(20),
  loc varchar(50)
);

insert into t_dept values (10,'testing','beijing');
insert into t_dept values (20,'saling','shanghai');
```

步骤二：新建工程，导入 jar 包

新建名为 SpringMyBatis_Day08_Part1 的 Web 工程，在该工程导入如图-1 所示的 jar 包：

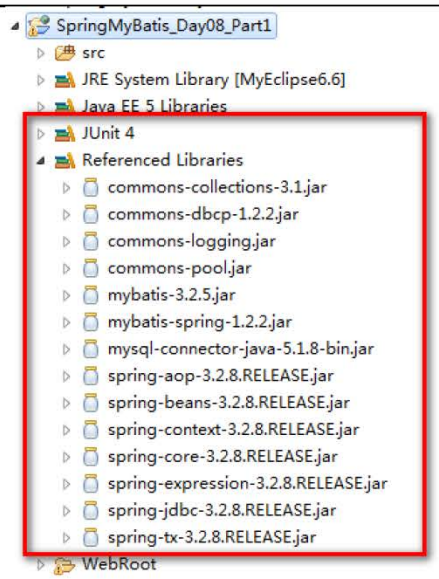


图-1

步骤三：新建 Dept 实体类

新建实体类 Dept，如图-2 所示：

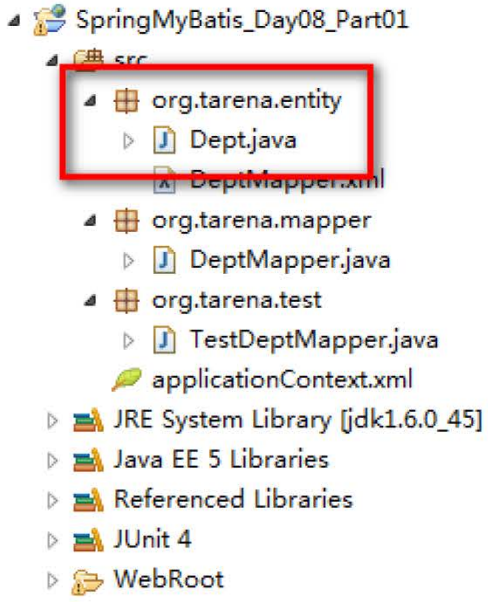


图-2

实体类 Dept 中的代码如下所示：

```
package org.tarena.entity;

public class Dept {
 private Integer deptno;
 private String dname;
 private String loc;

 public Integer getDeptno() {
     return deptno;
 }
```

```
    public void setDeptno(Integer deptno) {
        this.deptno = deptno;
    }

    public String getDname() {
        return dname;
    }

    public void setDname(String dname) {
        this.dname = dname;
    }

    public String getLoc() {
        return loc;
    }

    public void setLoc(String loc) {
        this.loc = loc;
    }
}
```

步骤四：新建 DeptMapper.xml 映射文件

新建映射文件 DeptMapper.xml，如图-3 所示：

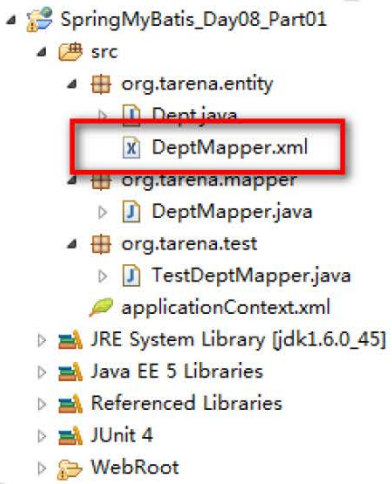


图-3

映射文件 DeptMapper.xml 的代码如下所示：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.mapper.DeptMapper">

 <select id="findAll" resultType="org.tarena.entity.Dept">
     select DEPTNO,DNAME,LOC from T_DEPT
 </select>

</mapper>
```

步骤五：新建 DeptMapper 映射接口

新建映射接口 DeptMapper，如图-4 所示：

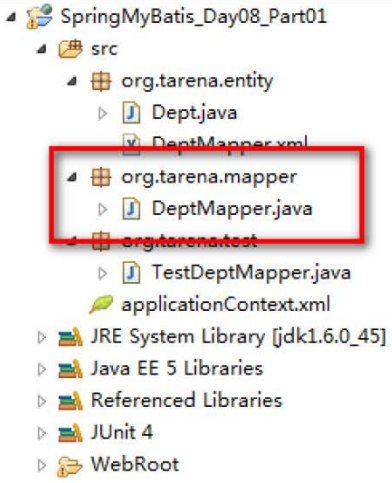


图-4

映射接口 DeptMapper 的代码如下所示：

```
package org.tarena.mapper;

import java.util.List;
import org.tarena.entity.Dept;

public interface DeptMapper {
 public List<Dept> findAll();
}
```

步骤六：新建 applicationContext.xml 配置文件

新建 spring 配置文件 applicationContext.xml，如图-5 所示：

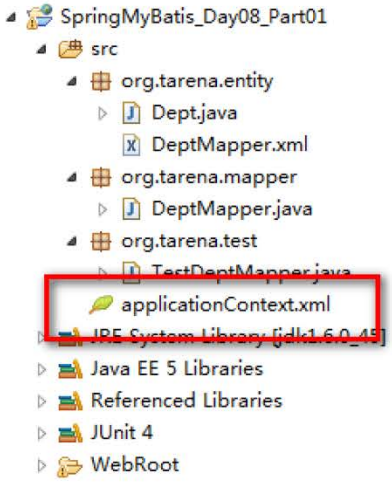


图-5

spring 配置文件 applicationContext.xml 的代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:jdbc="http://www.springframework.org/schema/jdbc"
 xmlns:jee="http://www.springframework.org/schema/jee"
```

```
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">
    <bean id="myDataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="root" />
    </bean>

    <bean id="sqlSessionFactory"
        class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="myDataSource" />
        <property name="mapperLocations"
            value="classpath:org/tarena/entity/*.xml" />
    </bean>
    <!-- 定义 Mapper -->
    <bean id="deptMapper"
        class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface"
            value="org.tarena.mapper.DeptMapper" />
        <property name="sqlSessionFactory" ref="sqlSessionFactory" />
    </bean>

   </beans>
```

步骤七：新建 TestDeptMapper 测试类

新建测试类 TestDeptMapper，如图-6 所示：

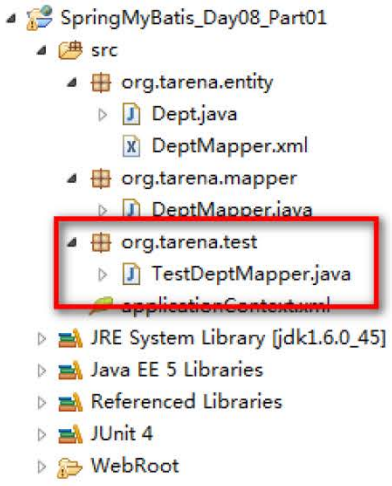


图-6

测试类 TestDeptMapper 的代码如下所示：

```
package org.tarena.test;

import java.io.IOException;
import java.util.List;

import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.dao.DeptDAO;
import org.tarena.entity.Dept;
import org.tarena.mapper.DeptMapper;

public class TestDeptMapper {
 @Test
 public void testFindAll() throws IOException {
     String conf = "applicationContext.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
     List<Dept> list = mapper.findAll();
     for (Dept dept : list) {
         System.out.println(dept.getDeptno() + " " + dept.getDname());
     }
 }
}
```

步骤八：运行 TestDeptMapper 测试类

运行测试类 TestDeptMapper，如图-7 所示：

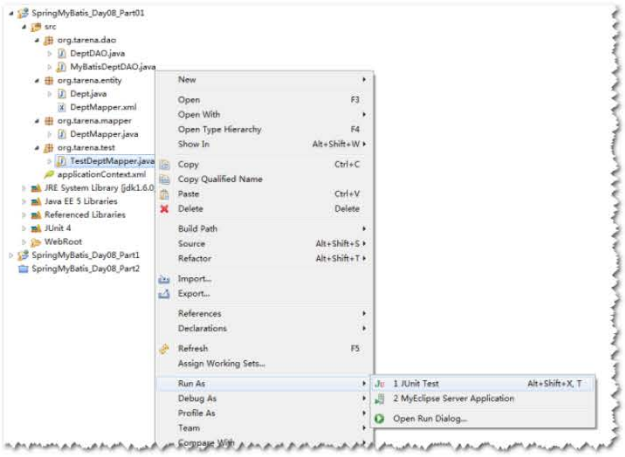


图-7

控制台输出方框内信息，说明测试成功，如图-8 所示：

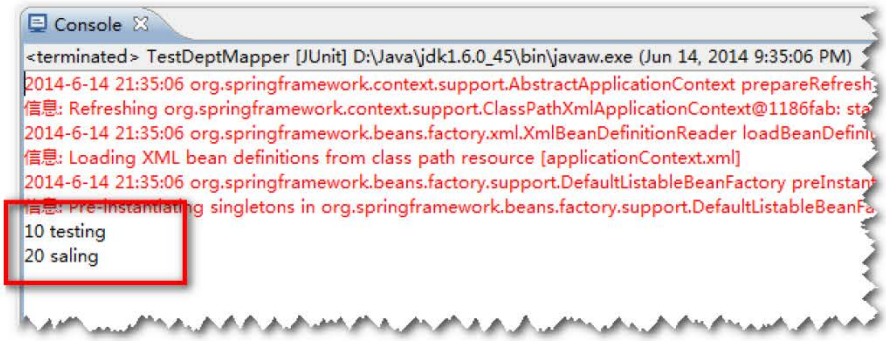


图-8

- **完整代码**

实体类 Dept 代码如下：

```
package org.tarena.entity;

public class Dept {
 private Integer deptno;
 private String dname;
 private String loc;

 public Integer getDeptno() {
     return deptno;
 }
 public void setDeptno(Integer deptno) {
     this.deptno = deptno;
 }
 public String getDname() {
     return dname;
 }
 public void setDname(String dname) {
     this.dname = dname;
 }
 public String getLoc() {
     return loc;
 }
 public void setLoc(String loc) {
     this.loc = loc;
 }
}
```

映射文件 DeptMapper.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.mapper.DeptMapper">

 <select id="findAll" resultType="org.tarena.entity.Dept">
     select DEPTNO,DNAME,LOC from T_DEPT
 </select>

</mapper>
```

映射接口 DeptMapper 代码如下：

```
package org.tarena.mapper;

import java.util.List;
import org.tarena.entity.Dept;

public interface DeptMapper {
 public List<Dept> findAll();
}
```

spring 配置文件 applicationContext.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:jee="http://www.springframework.org/schema/jee"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">
    <bean id="myDataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="root" />
    </bean>

    <bean id="sqlSessionFactory"
        class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="myDataSource" />
        <property name="mapperLocations"
            value="classpath:org/tarena/entity/*.xml" />
    </bean>
    <!-- 定义 Mapper -->
    <bean id="deptMapper"
        class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface"
            value="org.tarena.mapper.DeptMapper" />
        <property name="sqlSessionFactory" ref="sqlSessionFactory" />
    </bean>

   </beans>
```

测试类 TestDeptMapper 代码如下：

```
   package org.tarena.test;

   import java.io.IOException;
   import java.util.List;

   import org.junit.Test;
   import org.springframework.context.ApplicationContext;
   import org.springframework.context.support.ClassPathXmlApplicationContext;
   import org.tarena.dao.DeptDAO;
   import org.tarena.entity.Dept;
   import org.tarena.mapper.DeptMapper;

   public class TestDeptMapper {
    @Test
    public void testFindAll() throws IOException {
        String conf = "applicationContext.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
        List<Dept> list = mapper.findAll();
        for (Dept dept : list) {
```

```
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }
}
```

2. MapperScannerConfigurer 实现对 Dept 表查询操作案例

- **问题**

如何批量扫描 MyBatis 中的 Mapper 接口，可以简化一个 Mapper 接口一个 MapperFactoryBean 的定义方法？

- **方案**

MapperScannerConfigurer 可以批量扫描 MyBatis 的 Mapper 接口，并为每个接口生成一个 MapperFactoryBean 实例。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：新建 Spring 配置文件 applicationContext-scan.xml

新建 Spring 配置文件 applicationContext-scan.xml，如图-9 所示：

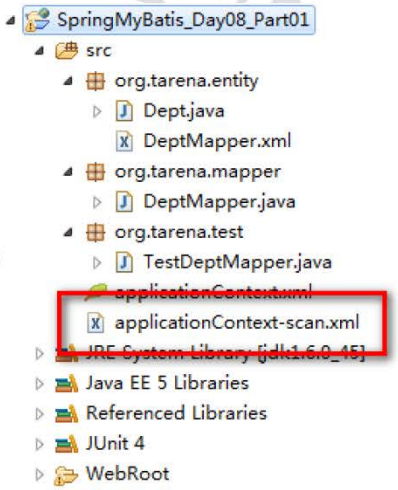


图-9

Spring 配置文件 applicationContext-scan.xml 中的代码如下所示：

```
    <?xml version="1.0" encoding="UTF-8"?>
    <beans                xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xsi:schemaLocation="
```

```
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">
    <bean id="myDataSource"
              class="org.apache.commons.dbcp.BasicDataSource"
              destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/test"/>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
    </bean>

    <bean                                                id="sqlSessionFactory1"
class="org.mybatis.spring.SqlSessionFactoryBean">
         <property name="dataSource" ref="myDataSource" />
         <property                                        name="mapperLocations"
value="classpath:org/tarena/entity/*.xml"/>
       </bean>

    <!-- Mapper 扫描 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
          <property name="basePackage" value="org.tarena.mapper" />
       </bean>

   </beans>
```

步骤二：修改 TestDeptMapper 测试类

修改 TestDeptMapper 测试类加入图-10 方框中的代码，如下图：

```
package org.tarena.test;

import java.io.IOException;

public class TestDeptMapper {

    @Test
    public void testFindAll() throws IOException {
        String conf = "applicationContext.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
        List<Dept> list = mapper.findAll();
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }

    @Test
    public void testScanFindAll() throws IOException {
        String conf = "applicationContext-scan.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
        List<Dept> list = mapper.findAll();
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }
}
```

图-10

TestDeptMapper 测试类加代码如下：

```
@Test
public void testScanFindAll() throws IOException {
    String conf = "applicationContext-scan.xml";
    ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
    DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
    List<Dept> list = mapper.findAll();
    for (Dept dept : list) {
        System.out.println(dept.getDeptno() + " " + dept.getDname());
    }
}
```

步骤三：运行 TestDeptMapper 测试

在 TestDeptMapper 类右键运行测试，如图-11 所示：

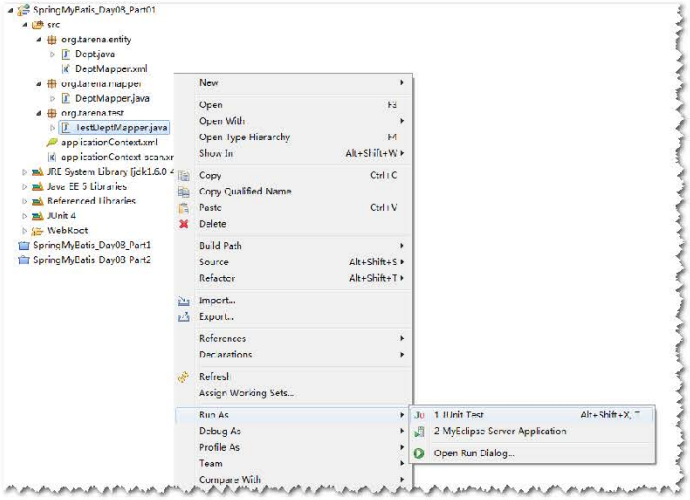


图-11

控制台输出方框内信息，说明测试成功。如图-12 所示：

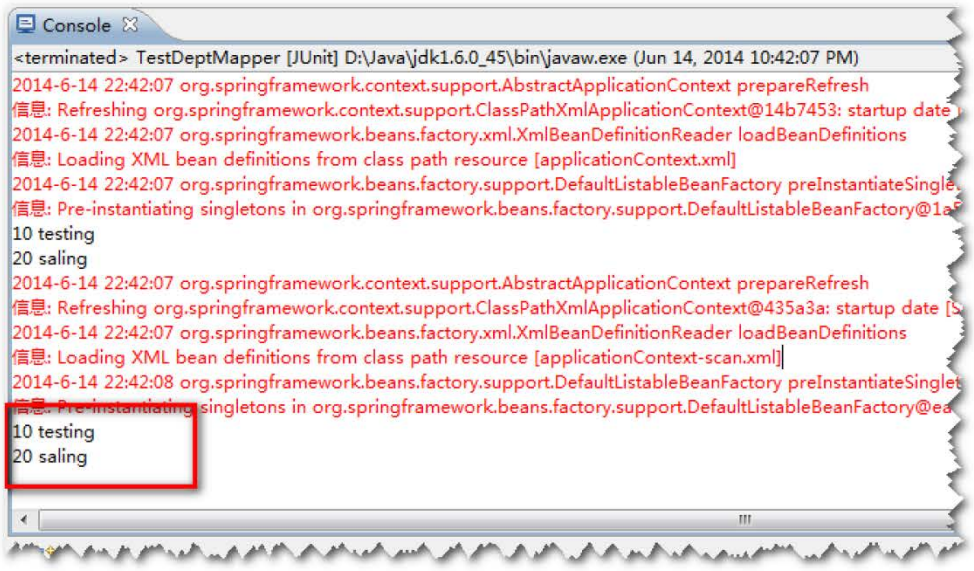


图-12

- **完整代码**

Spring 配置文件 applicationContext-scan.xml 代码如下：

```
   <?xml version="1.0" encoding="UTF-8"?>
   <beans                xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">
    <bean id="myDataSource"
              class="org.apache.commons.dbcp.BasicDataSource"
              destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/test"/>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
    </bean>

    <bean                                                id="sqlSessionFactory1"
class="org.mybatis.spring.SqlSessionFactoryBean">
         <property name="dataSource" ref="myDataSource" />
         <property                                        name="mapperLocations"
value="classpath:org/tarena/entity/*.xml"/>
       </bean>

    <!-- Mapper 扫描 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
          <property name="basePackage" value="org.tarena.mapper" />
       </bean>

   </beans>
```

测试类 TestDeptMapper 代码如下：

```
   package org.tarena.test;

   import java.io.IOException;
   import java.util.List;

   import org.junit.Test;
   import org.springframework.context.ApplicationContext;
   import org.springframework.context.support.ClassPathXmlApplicationContext;
   import org.tarena.entity.Dept;
   import org.tarena.mapper.DeptMapper;

   public class TestDeptMapper {

    @Test
```

```
    public void testFindAll() throws IOException {
        String conf = "applicationContext.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
        List<Dept> list = mapper.findAll();
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }

    @Test
    public void testScanFindAll() throws IOException {
        String conf = "applicationContext-scan.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
        List<Dept> list = mapper.findAll();
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }
}
```

3. 通过注解实现 MapperScannerConfigurer 对 Dept 表查询操作案例

- **问题**

将某个包中带有指定注解标记的 Mapper 接口扫描 ,转化成 MapperFactoryBean 对象。

- **方案**

可以先自定义一个注解标记，然后在 MapperScannerConfigurer 配置中将其指定给 annotationClass 属性。

这样在 Mapper 接口中带有自定义注解标记的就可以扫描生成 MapperFactoryBean，不带有注解标记的就不进行处理。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：新建 MyBatisRepository 注解类

新建注解类 MyBatisRepository，如图-13 所示：

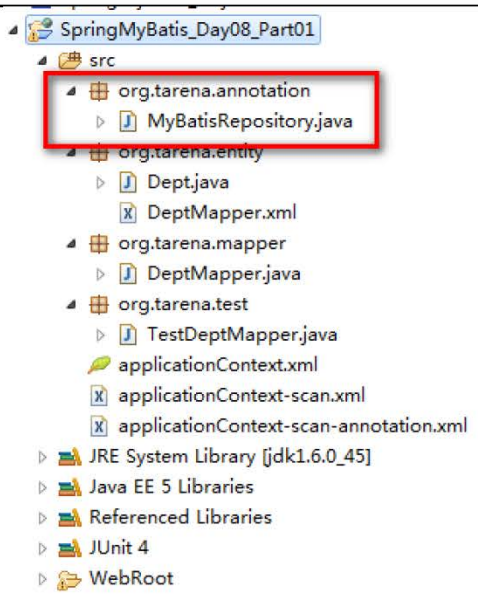


图-13

注解类 MyBatisRepository 的代码如下：

```
package org.tarena.annotation;

import org.springframework.stereotype.Repository;

@Repository
public @interface MyBatisRepository {
 String value() default "";
}
```

步骤二：修改 DeptMapper 映射接口

修改 DeptMapper 映射接口，加入图-14 方框内的注解代码：

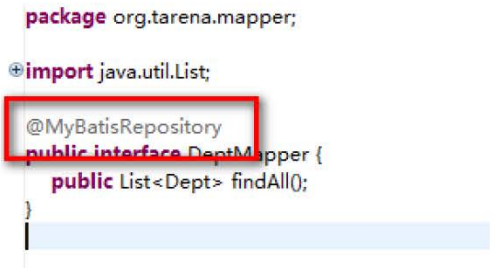


图-14

DeptMapper 映射接口加代码如下：

```
@MyBatisRepository
```

步骤三：新建 Spring 配置文件 applicationContext-scan-annotation.xml

新建 Spring 配置文件 applicationContext-scan-annotation.xml，如图-15 所示：

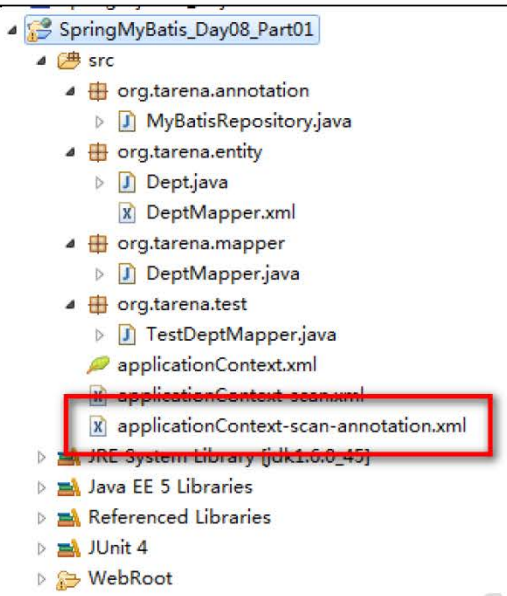


图-15

Spring 配置文件 applicationContext-scan-annotation.xml 中的代码如下所示：

```
    <?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:context="http://www.springframework.org/schema/context"
     xmlns:jdbc="http://www.springframework.org/schema/jdbc"
     xmlns:jee="http://www.springframework.org/schema/jee"
     xmlns:tx="http://www.springframework.org/schema/tx"
     xmlns:jpa="http://www.springframework.org/schema/data/jpa"
     xsi:schemaLocation="
         http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
         http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
         http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
         http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
         http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
         http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">
     <bean id="myDataSource"
         class="org.apache.commons.dbcp.BasicDataSource"
         destroy-method="close">
         <property name="driverClassName" value="com.mysql.jdbc.Driver" />
         <property name="url" value="jdbc:mysql://localhost:3306/test" />
         <property name="username" value="root" />
         <property name="password" value="root" />
     </bean>

     <bean id="sqlSessionFactory"
         class="org.mybatis.spring.SqlSessionFactoryBean">
         <property name="dataSource" ref="myDataSource" />
         <property name="mapperLocations"
             value="classpath:org/tarena/entity/*.xml" />
     </bean>

     <!-- 按指定包和注解标记扫描 Mapper -->
     <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
```

```
        <property name="sqlSessionFactory" ref="sqlSessionFactory" />
        <property name="basePackage" value="org.tarena" />
        <property name="annotationClass"
            value="org.tarena.annotation.MyBatisRepository" />
    </bean>

    </beans>
```

步骤四：修改 TestDeptMapper 测试类

修改 TestDeptMapper 测试类加入图-16 方框中的代码，如下图：

```
package org.tarena.test;

import java.io.IOException;

public class TestDeptMapper {

    @Test
    public void testFindAll() throws IOException {
        String conf = "applicationContext.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
        List<Dept> list = mapper.findAll();
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }

    @Test
    public void testScanFindAll() throws IOException {
        String conf = "applicationContext-scan.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
        List<Dept> list = mapper.findAll();
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }

    @Test
    public void testScanAnnotationFindAll() throws IOException {
        String conf = "applicationContext-scan-annotation.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
        List<Dept> list = mapper.findAll();
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }
}
```

图-16

TestDeptMapper 测试类加代码如下：

```
@Test
public void testScanAnnotationFindAll() throws IOException {
    String conf = "applicationContext-scan-annotation.xml";
    ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
    DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
    List<Dept> list = mapper.findAll();
    for (Dept dept : list) {
        System.out.println(dept.getDeptno() + " " + dept.getDname());
```

```
    }
}
```

步骤五：运行 TestDeptMapper 测试

在 TestDeptMapper 类右键运行测试，如图-17 所示：

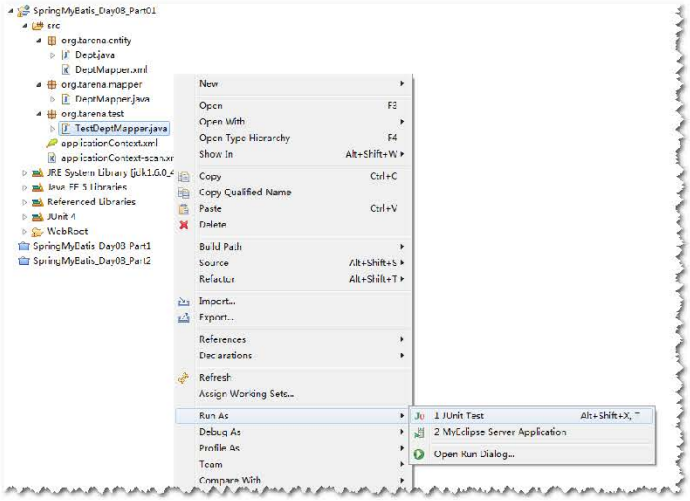


图-17

控制台输出方框内信息，说明测试成功。如图-18 所示：

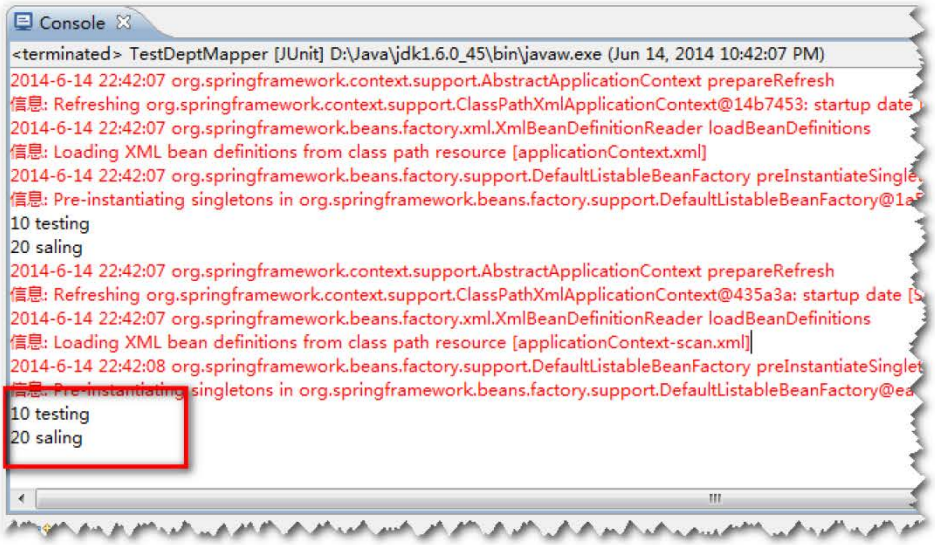


图-18

- **完整代码**

注解类 MyBatisRepository 代码如下：

```
package org.tarena.annotation;

import org.springframework.stereotype.Repository;
```

```
@Repository
public @interface MyBatisRepository {
 String value() default "";
}
```

映射接口 DeptMapper 代码如下：

```
package org.tarena.mapper;

import java.util.List;

import org.tarena.annotation.MyBatisRepository;
import org.tarena.entity.Dept;

@MyBatisRepository
public interface DeptMapper {
 public List<Dept> findAll();
}
```

Spring 配置文件 applicationContext-scan-annotation.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:jdbc="http://www.springframework.org/schema/jdbc"
 xmlns:jee="http://www.springframework.org/schema/jee"
 xmlns:tx="http://www.springframework.org/schema/tx"
 xmlns:jpa="http://www.springframework.org/schema/data/jpa"
 xsi:schemaLocation="
     http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
     http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
     http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
     http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
     http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
     http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">
 <bean id="myDataSource"
     class="org.apache.commons.dbcp.BasicDataSource"
     destroy-method="close">
     <property name="driverClassName" value="com.mysql.jdbc.Driver" />
     <property name="url" value="jdbc:mysql://localhost:3306/test" />
     <property name="username" value="root" />
     <property name="password" value="root" />
 </bean>

 <bean id="sqlSessionFactory"
     class="org.mybatis.spring.SqlSessionFactoryBean">
     <property name="dataSource" ref="myDataSource" />
     <property name="mapperLocations"
         value="classpath:org/tarena/entity/*.xml" />
 </bean>

 <!-- 按指定包和注解标记扫描 Mapper -->
 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
     <property name="sqlSessionFactory" ref="sqlSessionFactory" />
     <property name="basePackage" value="org.tarena" />
     <property name="annotationClass"
         value="org.tarena.annotation.MyBatisRepository" />
```

```
    </bean>
</beans>
```

测试类 TestDeptMapper 代码如下：

```
package org.tarena.test;

import java.io.IOException;
import java.util.List;

import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.entity.Dept;
import org.tarena.mapper.DeptMapper;

public class TestDeptMapper {
 @Test
 public void testFindAll() throws IOException {
     String conf = "applicationContext.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
     List<Dept> list = mapper.findAll();
     for (Dept dept : list) {
         System.out.println(dept.getDeptno() + " " + dept.getDname());
     }
 }

 @Test
 public void testScanFindAll() throws IOException {
     String conf = "applicationContext-scan.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
     List<Dept> list = mapper.findAll();
     for (Dept dept : list) {
         System.out.println(dept.getDeptno() + " " + dept.getDname());
     }
 }

 @Test
 public void testScanAnnotationFindAll() throws IOException {
     String conf = "applicationContext-scan-annotation.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
     List<Dept> list = mapper.findAll();
     for (Dept dept : list) {
         System.out.println(dept.getDeptno() + " " + dept.getDname());
     }
 }
}
```

4. 通过 SqlSessionTemplate 实现对 Dept 表的 Map 类型查询操作案例

- **问题**

在 Spring 和 MyBatis 整合应用中，除了使用 Mapper 接口注入外，还可以使用 SqlSessionTemplate 注入。

- **方案**

首先在 Spring 容器中定义一个 SqlSessionTemplate 对象，然后将它给 DAO 注入，在 DAO 中使用 SqlSessionTemplate 的方法完成增删改查操作。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：新建 DAO 接口 DeptDAO 和实现类 MyBatisDeptDAO

DAO 接口 DeptDAO 和实现类 MyBatisDeptDAO 如图-19 所示：

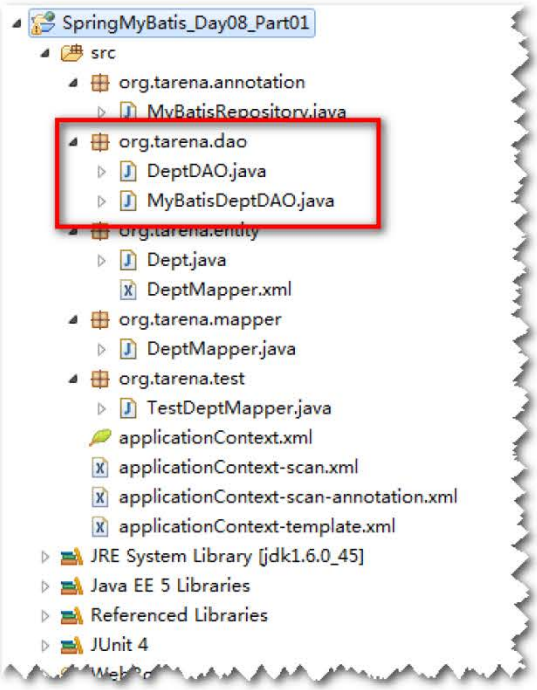


图-19

接口 DeptDAO 的代码如下：

```
package org.tarena.dao;

import java.util.List;

import org.tarena.entity.Dept;

public interface DeptDAO {
 public List<Dept> findAll();
}
```

接口 DeptDAO 实现类 MyBatisDeptDAO 的代码如下：

```
package org.tarena.dao;

import java.util.List;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.tarena.entity.Dept;
```

```
@Repository
public class MyBatisDeptDAO implements DeptDAO{

private SqlSessionTemplate template;

@Autowired
public void setTemplate(SqlSessionTemplate template) {
    this.template = template;
}

@Override
public List<Dept> findAll() {
    List<Dept> list = template.selectList("findAll");
    return list;
}
}
```

步骤二：新建 Spring 配置文件 applicationContext-template.xml

新建 Spring 配置文件 applicationContext-template.xml，如图-20 所示：

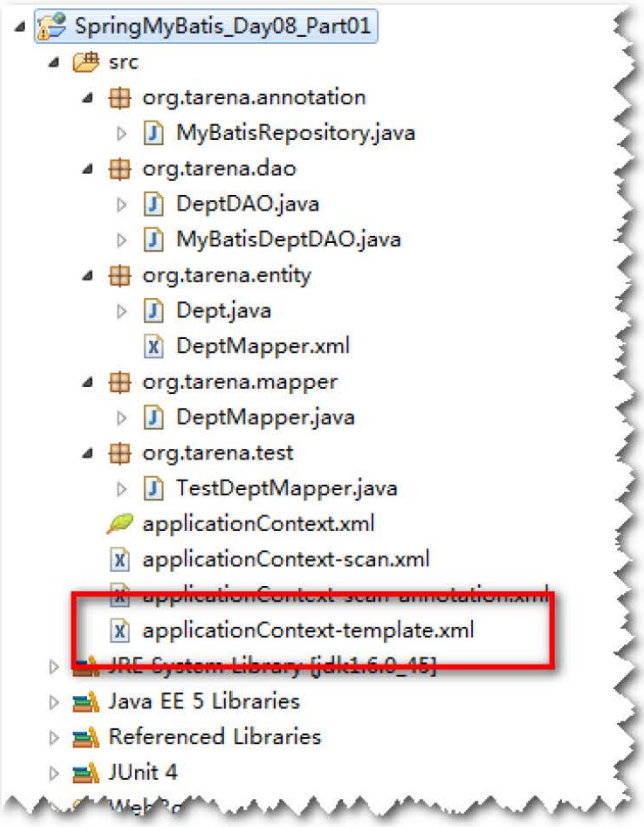


图-20

Spring 配置文件 applicationContext-template.xml 的代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans                    xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
```

```
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">
    <bean id="myDataSource"
              class="org.apache.commons.dbcp.BasicDataSource"
              destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/test"/>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
    </bean>

    <bean                                                id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="myDataSource" />
        <property                                        name="mapperLocations"
value="classpath:org/tarena/entity/*.xml"/>
      </bean>

    <!-- 定义 SqlSessionTemplate -->
    <bean                                              id="sqlSessionTemplate"
class="org.mybatis.spring.SqlSessionTemplate">
        <constructor-arg index="0" ref="sqlSessionFactory">
        </constructor-arg>
      </bean>

      <!-- 扫描 DAO 并注入 template -->
    <context:component-scan base-package="org.tarena.dao"/>
   </beans>
```

步骤三：修改 TestDeptMapper 测试类

修改 TestDeptMapper 测试类加入图-21 方框中的代码：

```
package org.tarena.test;

import java.io.IOException;

public class TestDeptMapper {

    public void testFindAll() throws IOException {

    public void testScanFindAll() throws IOException {

    public void testScanAnnotationFindAll() throws IOException {

    @Test
    public void testTemplateFindAll() throws IOException {
        String conf = "applicationContext-template.xml";
        ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
        DeptDAO dao = ac.getBean("myBatisDeptDAO", DeptDAO.class);
        List<Dept> list = dao.findAll();
        for (Dept dept : list) {
            System.out.println(dept.getDeptno() + " " + dept.getDname());
        }
    }
}
```

图-21

TestDeptMapper 测试类加代码如下：

```
@Test
public void testTemplateFindAll() throws IOException {
    String conf = "applicationContext-template.xml";
    ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
    DeptDAO dao = ac.getBean("myBatisDeptDAO", DeptDAO.class);
    List<Dept> list = dao.findAll();
    for (Dept dept : list) {
        System.out.println(dept.getDeptno() + " " + dept.getDname());
    }
}
```

步骤四：运行 TestDeptMapper 测试

在 TestDeptMapper 类右键运行测试，如图-22 所示：

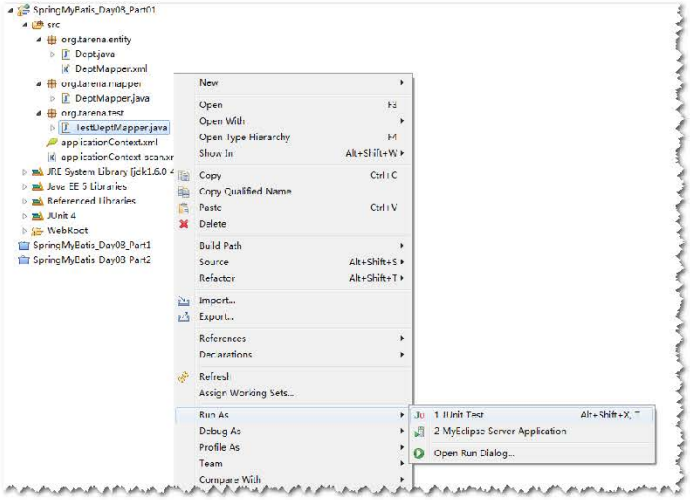


图-22

控制台输出方框内信息，说明测试成功。如图-23 所示：

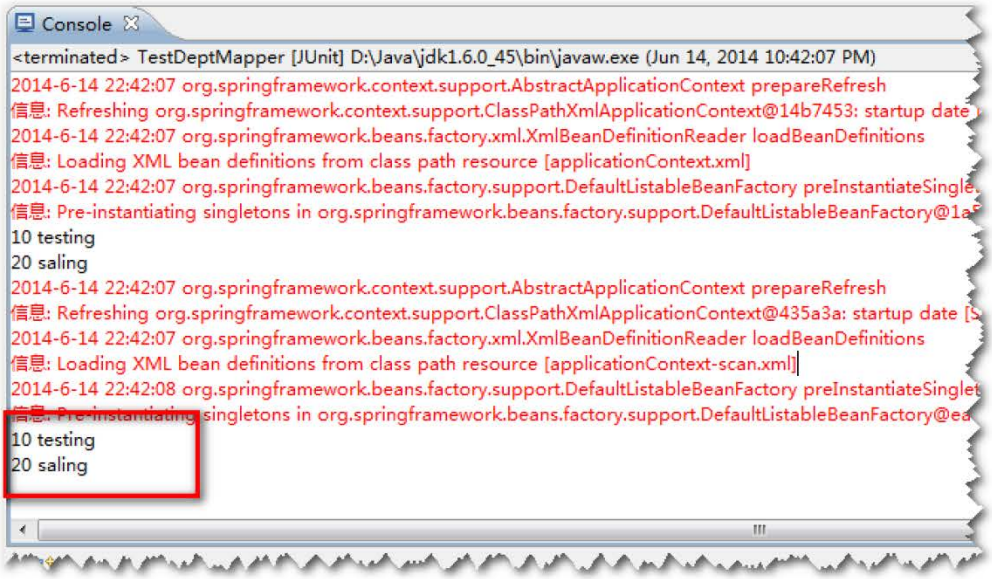


图-23

- **完整代码**

接口 DeptDAO 代码如下：

```
package org.tarena.dao;

import java.util.List;

import org.tarena.entity.Dept;

public interface DeptDAO {
 public List<Dept> findAll();
}
```

接口 DeptDAO 实现类 MyBatisDeptDAO 代码如下：

```
package org.tarena.dao;

import java.util.List;

import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.tarena.entity.Dept;

@Repository
public class MyBatisDeptDAO implements DeptDAO{

 private SqlSessionTemplate template;

 @Autowired
 public void setTemplate(SqlSessionTemplate template) {
     this.template = template;
 }

 @Override
 public List<Dept> findAll() {
     List<Dept> list = template.selectList("findAll");
     return list;
 }

}
```

Spring 配置文件 applicationContext-template.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans                 xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
```

```
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd">
    <bean id="myDataSource"
            class="org.apache.commons.dbcp.BasicDataSource"
            destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/test"/>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
    </bean>

    <bean                                                    id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="myDataSource" />
        <property                                            name="mapperLocations"
value="classpath:org/tarena/entity/*.xml"/>
      </bean>

    <!-- 定义 SqlSessionTemplate -->
    <bean                                                id="sqlSessionTemplate"
class="org.mybatis.spring.SqlSessionTemplate">
        <constructor-arg index="0" ref="sqlSessionFactory">
        </constructor-arg>
      </bean>

      <!-- 扫描 DAO 并注入 template -->
    <context:component-scan base-package="org.tarena.dao"/>
   </beans>
```

测试类 TestDeptMapper 代码如下：

```
   package org.tarena.test;

   import java.io.IOException;
   import java.util.List;

   import org.junit.Test;
   import org.springframework.context.ApplicationContext;
   import org.springframework.context.support.ClassPathXmlApplicationContext;
   import org.tarena.dao.DeptDAO;
   import org.tarena.entity.Dept;

   import org.tarena.mapper.DeptMapper;

   public class TestDeptMapper {

    @Test
    public void testFindAll() throws IOException {
    String conf = "applicationContext.xml";
    ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
    DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
    List<Dept> list = mapper.findAll();
    for (Dept dept : list) {
    System.out.println(dept.getDeptno() + " " + dept.getDname());
    }
    }

    @Test
    public void testScanFindAll() throws IOException {
    String conf = "applicationContext-scan.xml";
    ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
    DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
    List<Dept> list = mapper.findAll();
    for (Dept dept : list) {
```

```
    System.out.println(dept.getDeptno() + " " + dept.getDname());
    }
    }

    @Test
    public void testScanAnnotationFindAll() throws IOException {
    String conf = "applicationContext-scan-annotation.xml";
    ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
    DeptMapper mapper = ac.getBean("deptMapper", DeptMapper.class);
    List<Dept> list = mapper.findAll();
    for (Dept dept : list) {
    System.out.println(dept.getDeptno() + " " + dept.getDname());
    }
    }

   @Test
   public void testTemplateFindAll() throws IOException {
       String conf = "applicationContext-template.xml";
       ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
       DeptDAO dao = ac.getBean("myBatisDeptDAO", DeptDAO.class);
       List<Dept> list = dao.findAll();
       for (Dept dept : list) {
           System.out.println(dept.getDeptno() + " " + dept.getDname());
       }
   }
  }
```

5. 重构员工列表显示

- **问题**

利用 SpringMVC 和 MyBatis 实现员工列表显示页面。

- **方案**

首先使用 MapperScannerConfigurer 将带有@MyBatisRepository 注解的 DAO 接口扫描生成 MapperFactoryBean 实例。

然后通过MapperFactoryBean生成DAO实例给SpringMVC的Controller对象注入。

最后 Controller 将 DAO 查询返回的记录传递到 JSP 显示。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：新建工程，导入 jar 包

新建名为 SpringMyBatis_Day08_Part2 的 Web 工程,在该工程导入如图-24 所示的 jar 包：

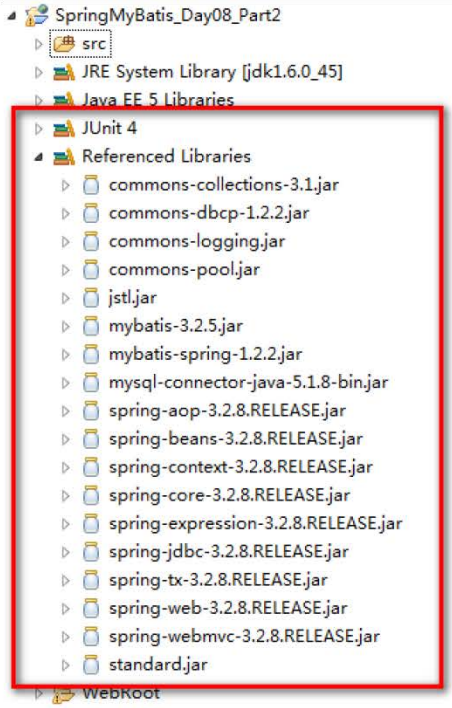


图-24

步骤二：新建 Emp 实体类和 EmpMapper.xml 映射文件

新建 Emp 实体类和 EmpMapper.xml 映射文件，如图-25 所示：

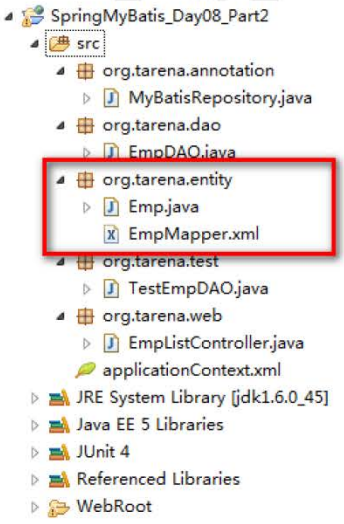


图-25

Emp 实体类的代码如下所示：

```
package org.tarena.entity;

import java.sql.Date;

public class Emp {
 private Integer empno;
 private String ename;
 private String job;
```

```
    private Integer mgr;
    private Date hiredate;
    private Double sal;
    private Double comm;
    private Integer deptno;

    public Integer getEmpno() {
        return empno;
    }
    public void setEmpno(Integer empno) {
        this.empno = empno;
    }
    public String getEname() {
        return ename;
    }
    public void setEname(String ename) {
        this.ename = ename;
    }
    public String getJob() {
        return job;
    }
    public void setJob(String job) {
        this.job = job;
    }
    public Integer getMgr() {
        return mgr;
    }
    public void setMgr(Integer mgr) {
        this.mgr = mgr;
    }
    public Date getHiredate() {
        return hiredate;
    }
    public void setHiredate(Date hiredate) {
        this.hiredate = hiredate;
    }
    public Double getSal() {
        return sal;
    }
    public void setSal(Double sal) {
        this.sal = sal;
    }
    public Double getComm() {
        return comm;
    }
    public void setComm(Double comm) {
        this.comm = comm;
    }
    public Integer getDeptno() {
        return deptno;
    }
    public void setDeptno(Integer deptno) {
        this.deptno = deptno;
    }
}
```

EmpMapper.xml 映射文件的代码如下所示：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
 "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
<mapper namespace="org.tarena.dao.EmpDAO">
 <select id="findAll" resultType="org.tarena.entity.Emp">
     select * from T_EMP
 </select>
</mapper>
```

步骤三：新建注解接口 MyBatisRepository

新建注解接口 MyBatisRepository，如图-26 所示：

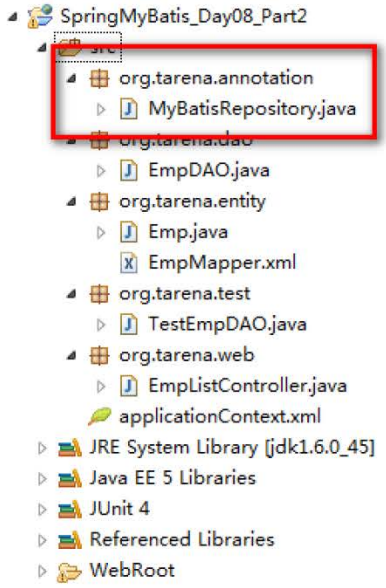


图-26

注解接口 MyBatisRepository 的代码如下：

```
package org.tarena.annotation;
import org.springframework.stereotype.Repository;

@Repository
public @interface MyBatisRepository {
 String value() default "";
}
```

步骤四：新建 DAO 接口 EmpDAO

新建 DAO 接口 EmpDAO，如图-27 所示：

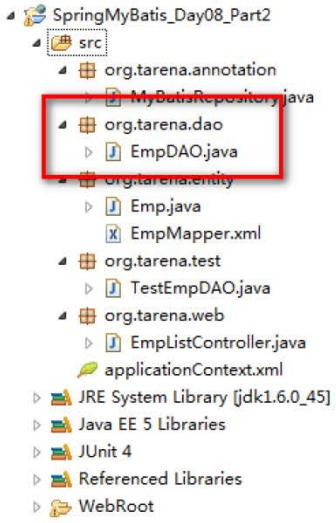


图-27

DAO 接口 EmpDAO 的代码如下所示：

```
package org.tarena.dao;

import java.util.List;

import org.tarena.annotation.MyBatisRepository;
import org.tarena.entity.Emp;

@MyBatisRepository
public interface EmpDAO {
 public List<Emp> findAll();
}
```

步骤五：新建 Spring 配置文件 applicationContext.xml

新建 Spring 配置文件 applicationContext.xml，如图-28 所示：

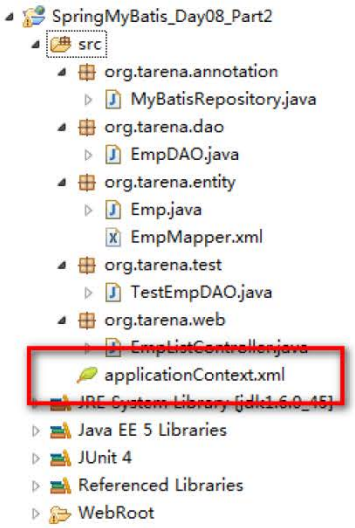


图-28

Spring 配置文件 applicationContext.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
```

```
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd
        http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">
    <bean id="myDataSource"
            class="org.apache.commons.dbcp.BasicDataSource"
            destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/test"/>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
    </bean>

    <bean                                                   id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="myDataSource" />
        <property                                           name="mapperLocations"
value="classpath:org/tarena/entity/*.xml"/>
      </bean>

    <!-- 按指定包和注解标记扫描 Mapper/DAO -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
          <property name="sqlSessionFactory" ref="sqlSessionFactory" />
          <property name="basePackage" value="org.tarena" />
          <property                                         name="annotationClass"
value="org.tarena.annotation.MyBatisRepository"/>
      </bean>

   </beans>
```

步骤六：新建 TestEmpDAO 测试类

新建 TestEmpDAO 测试类，如图-29 所示：

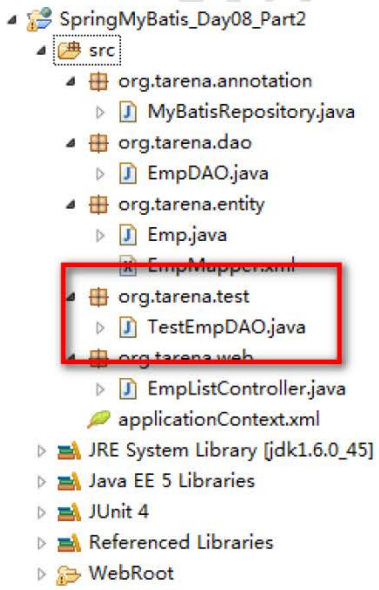


图-29

TestEmpDAO 测试类的代码如下所示：

```
   package org.tarena.test;

   import java.io.IOException;
   import java.util.List;
```

```
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

public class TestEmpDAO {
 @Test
 public void testFindAll() throws IOException{
     String conf = "applicationContext.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     EmpDAO mapper = ac.getBean("empDAO",EmpDAO.class);
     List<Emp> list = mapper.findAll();
     for(Emp emp : list){
         System.out.println(emp.getEmpno()+" "+emp.getEname());
     }
 }
}
```

步骤七：运行 TestEmpDAO 测试

在 TestEmpDAO 类右键运行测试，如图-30 所示：

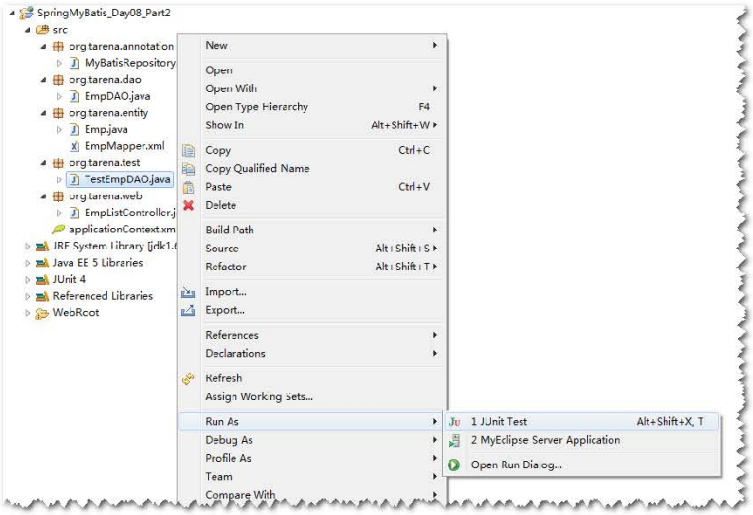


图-30

控制台输出方框内信息，说明测试成功。如图-31 所示：

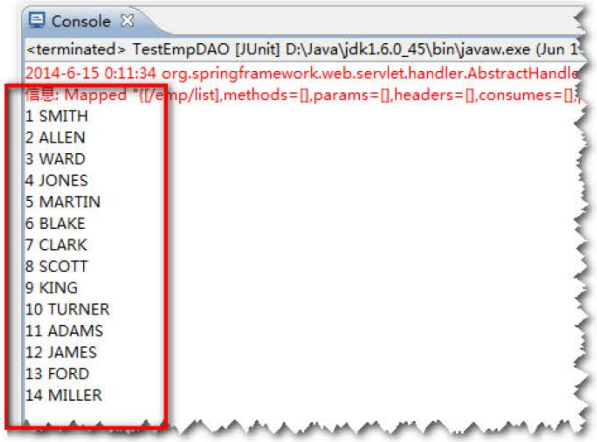


图-31

步骤八：新建 EmpListController 类

新建类 EmpListController，如图-32 所示：

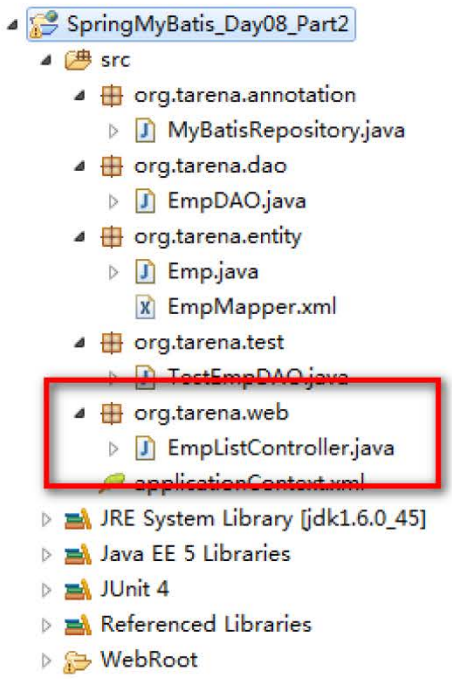


图-32

类 EmpListController 的代码如下所示：

```
package org.tarena.web;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

@Controller
@RequestMapping("emp")
public class EmpListController {
 private EmpDAO dao;
 @Autowired
 public void setDao(EmpDAO dao) {
     this.dao = dao;
 }
 @RequestMapping("/list")
 public String execute(Model model){
     List<Emp> list = dao.findAll();
     model.addAttribute("emps", list);
     return "emp_list";
 }
}
```

步骤九：新建 jsp 页面 emp_list.jsp

新建 jsp 页面 emp_list.jsp，如图-33 所示：

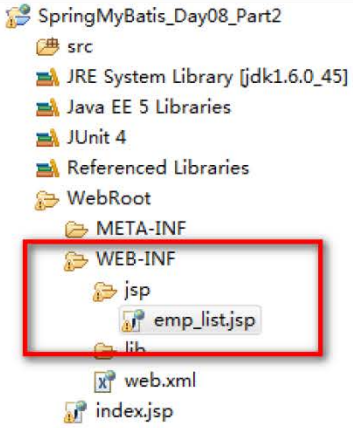


图-33

jsp 页面 emp_list.jsp 的代码如下所示：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
  <head>
    <title>员工列表示例</title>
  </head>
  <body>
    <table border="1">
     <tr>
         <td>编号</td>
         <td>姓名</td>
         <td>工资</td>
         <td>入职时间</td>
     </tr>
     <c:forEach items="${emps}" var="emp">
     <tr>
        <td>${emp.empno }</td>
         <td>${emp.ename }</td>
         <td>${emp.sal }</td>
         <td>${emp.hiredate }</td>
     </tr>
     </c:forEach>
    </table>
  </body>
</html>
```

步骤十：修改 web.xml 文件

修改 web.xml 内容，为下面的代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 <servlet>
     <servlet-name>SpringMVC</servlet-name>
     <servlet-class>
         org.springframework.web.servlet.DispatcherServlet
     </servlet-class>
     <init-param>
```

```
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:applicationContext.xml</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>SpringMVC</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

步骤十一：修改 Spring 配置文件 applicationContext.xml

修改 Spring 配置文件 applicationContext.xml，加入图-34 方框内的代码。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context" xmlns:jdbc="http://www.springframework.org/schema/jdbc"
  xmlns:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:jpa="http://www.springframework.org/schema/data/jpa" xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd
    http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
    http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
    http://www.springframework.org/schema/data/jpa http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd
    http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">
  <bean id="myDataSource"
  </bean>

  <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  </bean>

  <!-- 按指定包和注解标记扫描Mapper/DAO -->
  <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
  </bean>

  <!-- Sping MVC -->
  <context:component-scan base-package="org.tarena"/>

  <!-- 支持@RequestMapping请求和Controller映射 -->
  <mvc:annotation-driven/>

  <!-- SpringMVC的ViewResolver -->
  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property value="/WEB-INF/jsp/" name="prefix"/>
    <property value=".jsp" name="suffix"/>
  </bean>

</beans>
```

图-34

Spring 配置文件 applicationContext.xml 加入代码如下所示：

```
    <!-- Sping MVC -->
    <context:component-scan base-package="org.tarena"/>

    <!-- 支持@RequestMapping 请求和 Controller 映射 -->
    <mvc:annotation-driven/>

    <!-- SpringMVC 的 ViewResolver -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property value="/WEB-INF/jsp/" name="prefix"/>
    <property value=".jsp" name="suffix"/>
    </bean>
```

步骤十二：Web 方式运行 SpringMyBatis_Day08_Part2 项目

在 Tomcat 中运行 SpringMyBatis_Day08_Part2 项目，如图-35 所示：

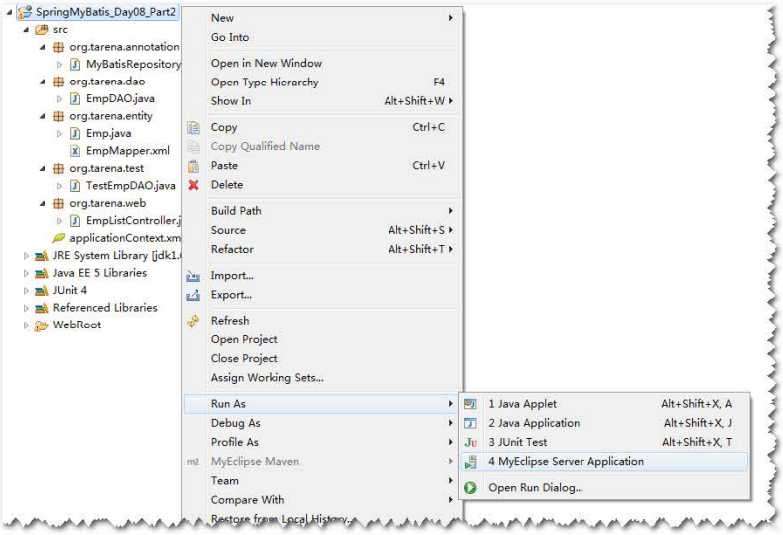
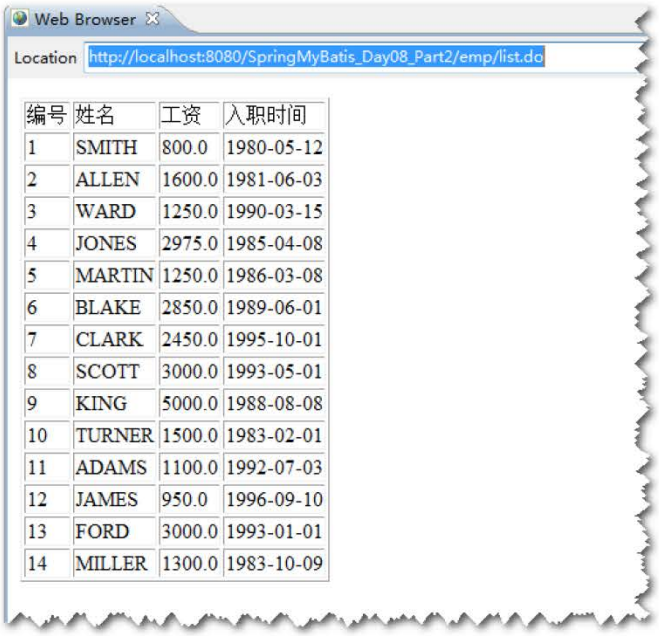


图-35

在浏览器输入访问地址：
http://localhost:8080/SpringMyBatis_Day08_Part2/emp/list.do
看到以下信息，说明测试成功。如图-36 所示：



编号	姓名	工资	入职时间
1	SMITH	800.0	1980-05-12
2	ALLEN	1600.0	1981-06-03
3	WARD	1250.0	1990-03-15
4	JONES	2975.0	1985-04-08
5	MARTIN	1250.0	1986-03-08
6	BLAKE	2850.0	1989-06-01
7	CLARK	2450.0	1995-10-01
8	SCOTT	3000.0	1993-05-01
9	KING	5000.0	1988-08-08
10	TURNER	1500.0	1983-02-01
11	ADAMS	1100.0	1992-07-03
12	JAMES	950.0	1996-09-10
13	FORD	3000.0	1993-01-01
14	MILLER	1300.0	1983-10-09

图-36

- **完整代码**

实体类 Emp 代码如下：

```
package org.tarena.entity;

import java.sql.Date;
```

```
public class Emp {
 private Integer empno;
 private String ename;
 private String job;
 private Integer mgr;
 private Date hiredate;
 private Double sal;
 private Double comm;
 private Integer deptno;

 public Integer getEmpno() {
     return empno;
 }

 public void setEmpno(Integer empno) {
     this.empno = empno;
 }

 public String getEname() {
     return ename;
 }

 public void setEname(String ename) {
     this.ename = ename;
 }

 public String getJob() {
     return job;
 }

 public void setJob(String job) {
     this.job = job;
 }

 public Integer getMgr() {
     return mgr;
 }

 public void setMgr(Integer mgr) {
     this.mgr = mgr;
 }

 public Date getHiredate() {
     return hiredate;
 }

 public void setHiredate(Date hiredate) {
     this.hiredate = hiredate;
 }

 public Double getSal() {
     return sal;
 }

 public void setSal(Double sal) {
     this.sal = sal;
 }

 public Double getComm() {
     return comm;
 }

 public void setComm(Double comm) {
     this.comm = comm;
 }

 public Integer getDeptno() {
     return deptno;
```

```
    }

    public void setDeptno(Integer deptno) {
        this.deptno = deptno;
    }

    }
```

映射文件 EmpMapper.xml 代码如下：

```
    <?xml version="1.0" encoding="UTF-8" ?>
    <!DOCTYPE mapper PUBLIC "-//ibatis.apache.org//DTD Mapper 3.0//EN"
     "http://ibatis.apache.org/dtd/ibatis-3-mapper.dtd">
    <mapper namespace="org.tarena.dao.EmpDAO">

     <select id="findAll" resultType="org.tarena.entity.Emp">
         select * from T EMP
     </select>

    </mapper>
```

注解接口 MyBatisRepository 代码如下：

```
    package org.tarena.annotation;

    import org.springframework.stereotype.Repository;

    @Repository
    public @interface MyBatisRepository {
     String value() default "";
    }
```

DAO 接口 EmpDAO 代码如下：

```
    package org.tarena.dao;

    import java.util.List;

    import org.tarena.annotation.MyBatisRepository;
    import org.tarena.entity.Emp;

    @MyBatisRepository
    public interface EmpDAO {
     public List<Emp> findAll();
    }
```

Spring 配置文件 applicationContext.xml 代码如下：

```
    <?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:context="http://www.springframework.org/schema/context"
     xmlns:jdbc="http://www.springframework.org/schema/jdbc"
     xmlns:jee="http://www.springframework.org/schema/jee"
     xmlns:tx="http://www.springframework.org/schema/tx"
     xmlns:jpa="http://www.springframework.org/schema/data/jpa"
     xmlns:mvc="http://www.springframework.org/schema/mvc"
     xsi:schemaLocation="
         http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
```

```
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa-1.3.xsd
        http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">
    <bean id="myDataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/test" />
        <property name="username" value="root" />
        <property name="password" value="root" />
    </bean>

    <bean id="sqlSessionFactory"
        class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="myDataSource" />
        <property name="mapperLocations"
            value="classpath:org/tarena/entity/*.xml" />
    </bean>

    <!-- 按指定包和注解标记扫描 Mapper/DAO -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="sqlSessionFactory" ref="sqlSessionFactory" />
        <property name="basePackage" value="org.tarena" />
        <property name="annotationClass"
            value="org.tarena.annotation.MyBatisRepository" />
    </bean>

    <!-- Sping MVC -->
    <context:component-scan base-package="org.tarena.web" />

    <!-- 支持@RequestMapping 请求和 Controller 映射 -->
    <mvc:annotation-driven />

    <!-- SpringMVC 的 ViewResolver -->
    <bean

    class="org.springframework.web.servlet.view.InternalResourceViewResolver"
>
        <property value="/WEB-INF/jsp/" name="prefix" />
        <property value=".jsp" name="suffix" />
    </bean>

   </beans>
```

测试类 TestEmpDAO 代码如下：

```
   package org.tarena.test;

   import java.io.IOException;
   import java.util.List;

   import org.junit.Test;
   import org.springframework.context.ApplicationContext;
   import org.springframework.context.support.ClassPathXmlApplicationContext;
   import org.tarena.dao.EmpDAO;
   import org.tarena.entity.Emp;
```

```
public class TestEmpDAO {
 @Test
 public void testFindAll() throws IOException{
     String conf = "applicationContext.xml";
     ApplicationContext ac = new ClassPathXmlApplicationContext(conf);
     EmpDAO mapper = ac.getBean("empDAO",EmpDAO.class);
     List<Emp> list = mapper.findAll();
     for(Emp emp : list){
         System.out.println(emp.getEmpno()+" "+emp.getEname());
     }
 }
}
```

类 EmpListController 代码如下：

```
package org.tarena.web;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.tarena.dao.EmpDAO;
import org.tarena.entity.Emp;

@Controller
@RequestMapping("emp")
public class EmpListController {
 private EmpDAO dao;
 @Autowired
 public void setDao(EmpDAO dao) {
     this.dao = dao;
 }
 @RequestMapping("/list")
 public String execute(Model model){
     List<Emp> list = dao.findAll();
     model.addAttribute("emps", list);
     return "emp_list";
 }
}
```

jsp 页面 emp_list.jsp 代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
  <head>
    <title>员工列表示例</title>
  </head>

  <body>
    <table border="1">
      <tr>
          <td>编号</td>
          <td>姓名</td>
          <td>工资</td>
          <td>入职时间</td>
      </tr>
      <c:forEach items="${emps}" var="emp">
      <tr>
         <td>${emp.empno }</td>
          <td>${emp.ename }</td>
```

```
            <td>${emp.sal }</td>
            <td>${emp.hiredate }</td>
        </tr>
        </c:forEach>
      </table>
    </body>
  </html>
```

web.xml 代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 <servlet>
     <servlet-name>SpringMVC</servlet-name>
     <servlet-class>
         org.springframework.web.servlet.DispatcherServlet
     </servlet-class>
     <init-param>
         <param-name>contextConfigLocation</param-name>
         <param-value>classpath:applicationContext.xml</param-value>
     </init-param>
 </servlet>

 <servlet-mapping>
     <servlet-name>SpringMVC</servlet-name>
     <url-pattern>*.do</url-pattern>
 </servlet-mapping>

 <welcome-file-list>
     <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>
</web-app>
```

课后作业

1. 利用 Spring+MyBatis 实现员工更新功能。