* 学习目标

 * 能够掌握Mybatis 高级结果映射

  * 表的列名和类的属性名字不一致的情况

  * ResultMap--Result--column--property

 * 能够掌握MyBatis操作多表的关系

   * 一对多

      collection

   * 多对多

      collection

 * 能够掌握MyBatis延迟加载

----

 * 回顾

  * mybatis：

   * 持久层的框架，主要用于操作数据库，前生ibatis

   * JDBC存在数据库链接硬编码，SQL语句写在代码里存在硬编码,数据库连接池方面

   * 添加依赖，全局配置文件（SqlMapConfig.xml)，xml配置

   * SqlSessionFactoryBuilder---build(is)---SqlSessionFacatory---openSession--->MySqlSessionUtils

   * SqlSession--selectList(namespac+"."+id,args)

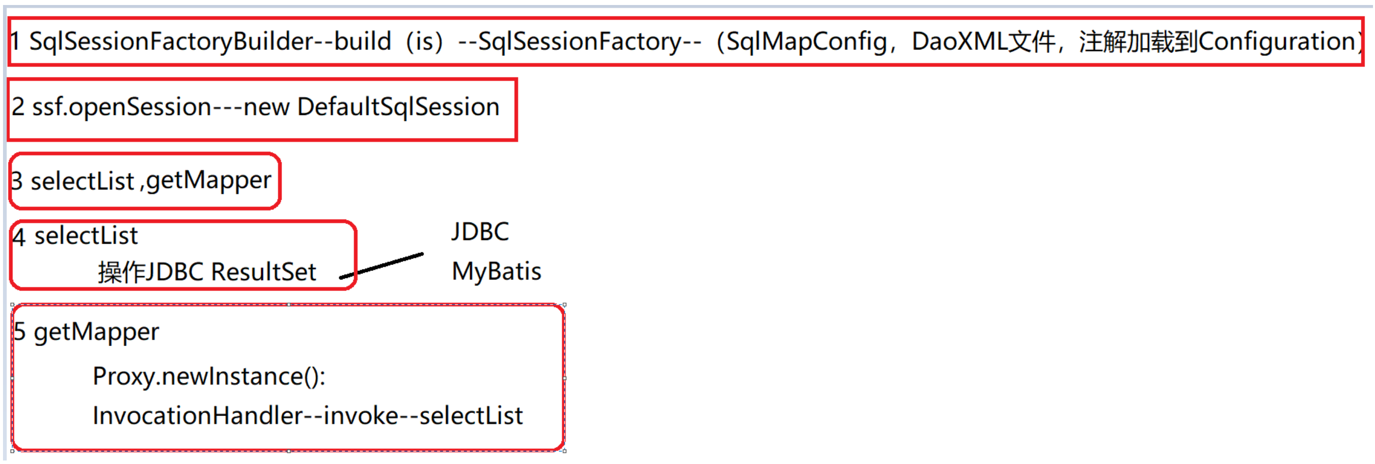
   * sqlSession.close

   * UserDao---UserDaoImpl

   * SqlSession--<T> T getMapper(Class clazz)---Proxy.newInstance(classLoader,interfaces[],InvocationHandler);

   * xml配置,注解

 * 源码阅读总结



1 SqlSessionFactoryBuilder--build（is）--SqlSessionFactory--（SqlMapConfig，DaoXML文件，注解加载到Configuration）

2 ssf.openSession---new DefaultSqlSession

3 selectList ,getMapper

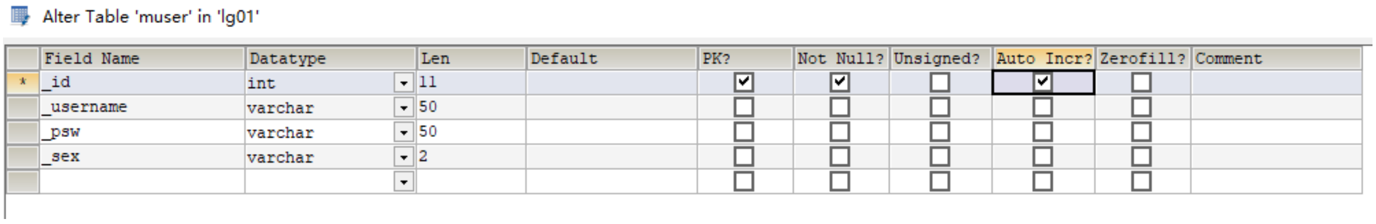
4 selectList
操作JDBC ResultSet

JDBC
MyBatis

5 getMapper
Proxy.newInstance():
InvocationHandler--invoke--selectList

* 能够掌握Mybatis 高级结果映射

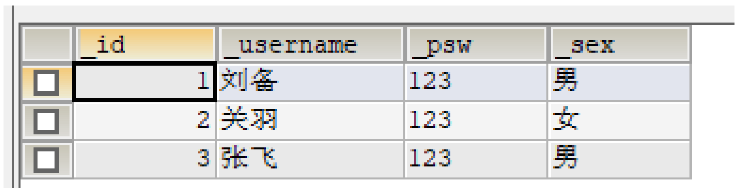
* 表的字段名与类的属性名字不一致情况



Alter Table 'muser' in 'lg01'

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
_id	int	11		✓	✓	☐	✓	☐	
_username	varchar	50		☐	☐	☐	☐	☐	
_psw	varchar	50		☐	☐	☐	☐	☐	
_sex	varchar	2		☐	☐	☐	☐	☐	
				☐	☐	☐	☐	☐	

```sql
INSERT INTO muser(_username,_psw,_sex) VALUES("刘备","123","男");
INSERT INTO muser(_username,_psw,_sex) VALUES("关羽","123","女");
INSERT INTO muser(_username,_psw,_sex) VALUES("张飞","123","男");
SELECT * FROM muser;
```

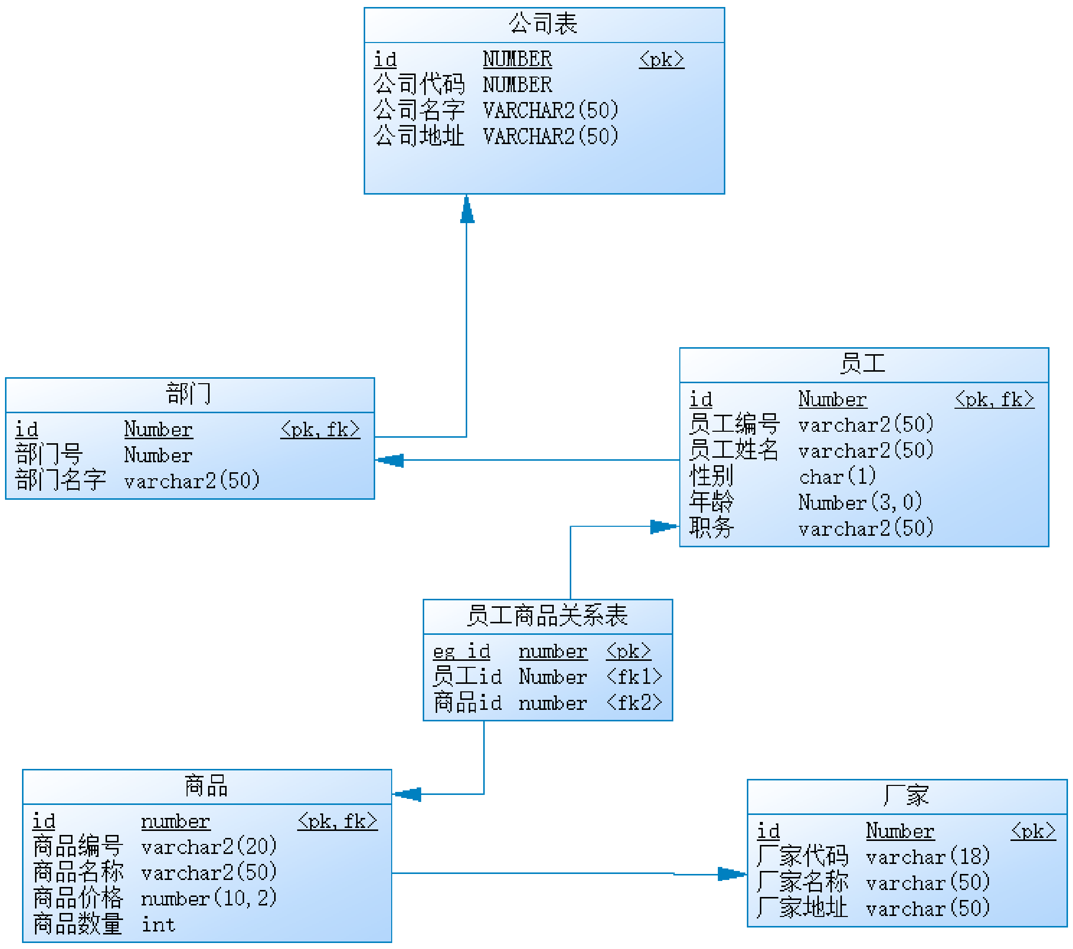


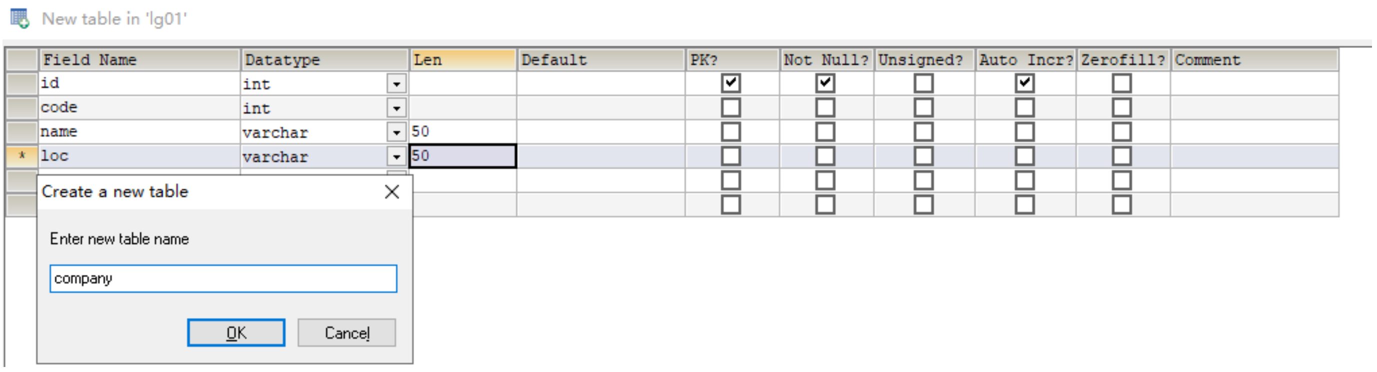
_id	_username	_psw	_sex
1	刘备	123	男
2	关羽	123	女
3	张飞	123	男

```java
* 案例（属性名和表名不一致）
* 配置xml的形式
public interface UserDao3 {
    /**
     * @param id
     * @return
     */
    public User findUserById(int id);
```

```
}
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.lg.dao.UserDao3">

    <!-- 第一种方式-->
    <!--<select id="findUserById" parameterType="int" resultType="com.lg.bean.U
        SELECT _id id,_username username,_psw psw,_sex sex FROM muser WHERE _id
    </select>-->
    <!-- 第二种方式-->
    <resultMap id="rstMap" type="com.lg.bean.User">
        <id column="_id" property="id" />
        <result column="_username" property="username"/>
        <result column="_psw" property="psw"/>
        <result column="_sex" property="sex"/>
    </resultMap>

    <select id="findUserById" parameterType="int" resultMap="rstMap">
        SELECT _id,_username,_psw,_sex FROM muser WHERE _id=#{id};
    </select>
</mapper>
* SqlMapConfig配置
 * <mapper resource="com/lg/dao/UserDao3"></mapper>
 @Test
public void test6(){
        SqlSession session = MyBatisUtils.getSqlSession();
        UserDao3 userDao = session.getMapper(UserDao3.class);
        User user = userDao.findUserById(1);
        System.out.println(user);
        session.close();
}
* 注解的形式
public interface UserDao4 {
    /**
     * @param id
     * @return
     */
//    @Select("SELECT _id id,_username username,_psw psw,_sex sex FROM muser WH
    @Select("SELECT _id ,_username,_psw,_sex FROM muser WHERE _id=#{id}")
    @Results({
            @Result(column="_id", property="id", id=true),
            @Result(column="_username", property="username"),
            @Result(column="_psw", property="psw"),
            @Result(column="_sex", property="sex")})
    public User findUserById(int id);
}
* SqlMapConfig配置
 <mapper class="com.lg.dao.UserDao4"></mapper>
 * 单元测试
```

* 能够掌握MyBatis操作多表的关系




New table in 'lg01'

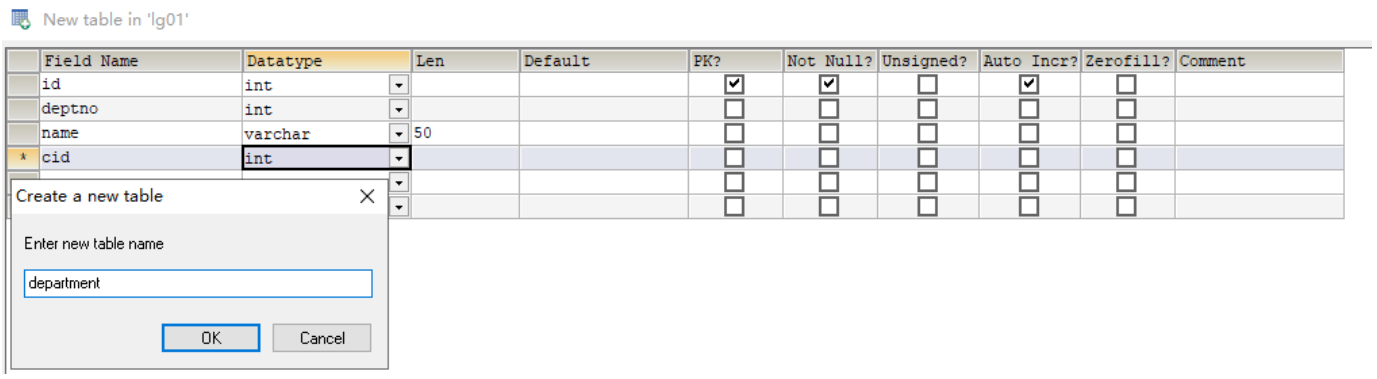
	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
	id	int			✓	✓		✓		
	code	int								
	name	varchar	50							
*	loc	varchar	50							

Create a new table

Enter new table name

company

OK Cancel


New table in 'lg01'

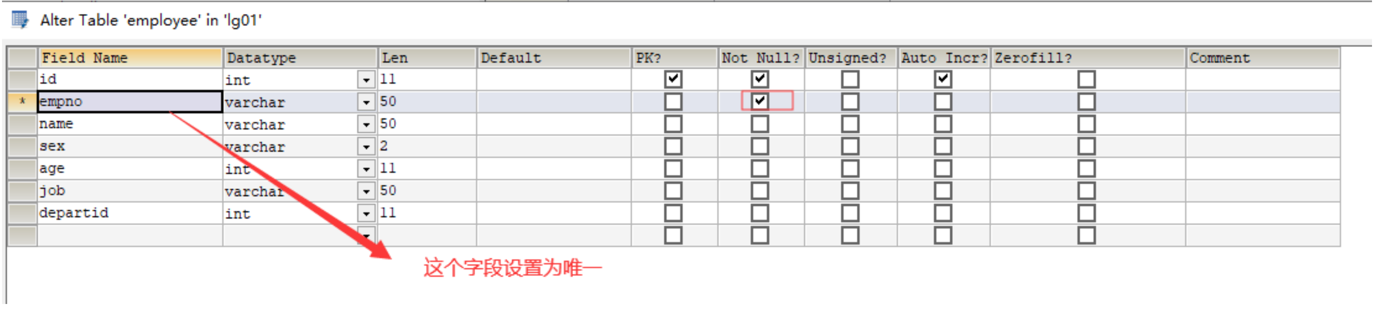
	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
	id	int			✓	✓		✓		
	deptno	int								
	name	varchar	50							
*	cid	int								

Create a new table

Enter new table name

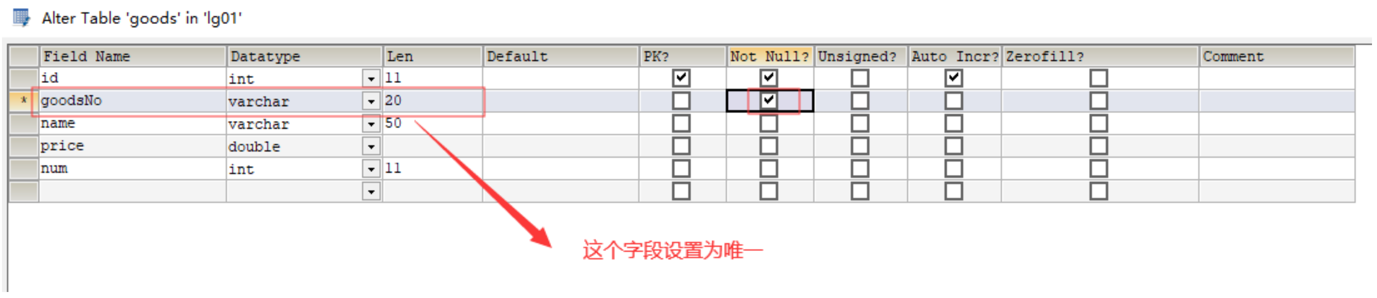
department

OK Cancel


Alter Table 'employee' in 'lg01'

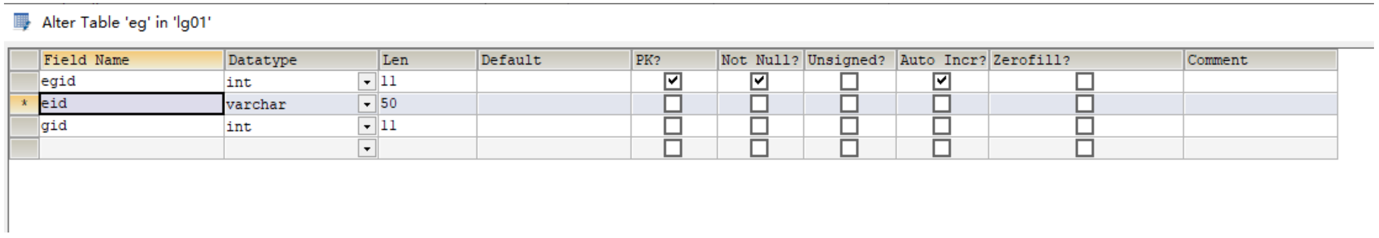
	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
	id	int	11		✓	✓		✓		
*	empno	varchar	50			✓				
	name	varchar	50							
	sex	varchar	2							
	age	int	11							
	job	varchar	50							
	departid	int	11							

这个字段设置为唯一


Alter Table 'goods' in 'lg01'

	Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
	id	int	11		✓	✓		✓		
*	goodsNo	varchar	20			✓				
	name	varchar	50							
	price	double								
	num	int	11							

这个字段设置为唯一


Alter Table 'eg' in 'lg01'

Field Name	Datatype	Len	Default	PK?	Not Null?	Unsigned?	Auto Incr?	Zerofill?	Comment
egid	int	11		✓	✓	☐	✓	☐	
eid	varchar	50		☐	☐	☐	☐	☐	
gid	int	11		☐	☐	☐	☐	☐	
				☐	☐	☐	☐	☐	

```
* 插入数据
INSERT INTO company(CODE,NAME,loc) VALUES(1001,'亮哥教育','广州');
INSERT INTO department(deptno,NAME,cid) VALUES(1,'教学部',1001);
INSERT INTO department(deptno,NAME,cid) VALUES(2,'研发部',1001);
INSERT INTO department(deptno,NAME,cid) VALUES(3,'渠道部',1001);
INSERT INTO employee(empno,NAME,sex,age,job,departid) VALUES('LG001','亮哥','男'
INSERT INTO employee(empno,NAME,sex,age,job,departid) VALUES('LG002','小黑','男'
INSERT INTO employee(empno,NAME,sex,age,job,departid) VALUES('LG003','小白','女'
INSERT INTO employee(empno,NAME,sex,age,job,departid) VALUES('LG008','老刘','男'
INSERT INTO goods(goodsNo,NAME,price,num) VALUES(10001,'Java教学视频','99',1000)
INSERT INTO goods(goodsNo,NAME,price,num) VALUES(10002,'MyBatis教学视频','199',1
INSERT INTO goods(goodsNo,NAME,price,num) VALUES(10003,'SpringBoot教学视频','29
INSERT INTO eg(eid,gid) VALUES('LG001',10001);
INSERT INTO eg(eid,gid) VALUES('LG001',10002);
INSERT INTO eg(eid,gid) VALUES('LG008',10003);
INSERT INTO eg(eid,gid) VALUES('LG008',10001);

* 温馨提醒：sqlyog设置唯一约束
  选中你要改的表--->右键-->选"Manage Indexes"--->点"NEW"-->建立索引时在"Index Opti
```

* 实体的准备：

```
 * Company(公司)，Department（部门），Empoyee（员工），Goods（商品）
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Company {
    private int id;
    private int code;
    private String name;
```

```java
    private String loc;
    private List<Department> departments;
}
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Department {
    private int id;
    private int deptno;
    private String name;
    private List<Employee> employees;
}
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Employee {
    private int id;
    private String empno;
    private String name;
    private String sex;
    private int age;
    private String job;
    private List<Goods> goods;
}
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Goods {
    private int id;
    private String goodsNo;
    private String name;
    private double price;
    private int num;
    private List<Employee> employees;
}
```

* 一对多（多对一）

```
* 案例：通过公司编号1001查询所有该公司的部门
```

```
 * 方式一
  * sql
SELECT * FROM company c WHERE c.code=1001;
SELECT * FROM department d WHERE d.cid=1001;
  * Dao
public interface CompanyDao {
    /**
     * @param code
     * @return
     * 通过公司编号获取Company
     */
    public Company getCompany(int code);
}

  * 配置文件
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.lg.dao.CompanyDao">
    <select id="getCompany" parameterType="int" resultMap="rst">
        SELECT * FROM company c WHERE c.code=#{code}
    </select>
    <resultMap id="rst" type="com.lg.bean.Company">
        <collection property="departments" ofType="com.lg.bean.Department" colu
    </resultMap>
    <select id="getDepartments" parameterType="int" resultType="com.lg.bean.Dep
        SELECT * FROM department d WHERE d.cid=#{cid};
    </select>
</mapper>
 * 单元测试
 @Test
 public void test1(){
        SqlSession sqlSession = MyBatisUtils.getSqlSession();
        CompanyDao companyDao = sqlSession.getMapper(CompanyDao.class);
        Company company = companyDao.getCompany(1001);
        System.out.println(company);
        sqlSession.close();
 }
 * 方式二
 * sql
 SELECT c.id cid,c.code ccode,c.name cname,c.loc cloc,d.id did,d.deptno ddeptn
 FROM company c,department d WHERE c.code=d.cid AND c.code=1001
 * Dao 一样
 * 配置文件
 <select id="getCompany" parameterType="int" resultMap="rst">
       SELECT c.id cid,c.code ccode,c.name cname,c.loc cloc,d.id did,d.deptno
       FROM company c,department d WHERE c.code=d.cid AND c.code=#{code}
 </select>
 <resultMap id="rst" type="com.lg.bean.Company">
      <result column="cid" property="id"></result>
      <result column="ccode" property="code"></result>
      <result column="cname" property="name"></result>
      <result column="cloc" property="loc"></result>
      <collection property="departments" ofType="com.lg.bean.Department">
           <result column="did" property="id"></result>
           <result column="ddeptno" property="deptno"></result>
           <result column="dname" property="name"></result>
      </collection>
 </resultMap>

```

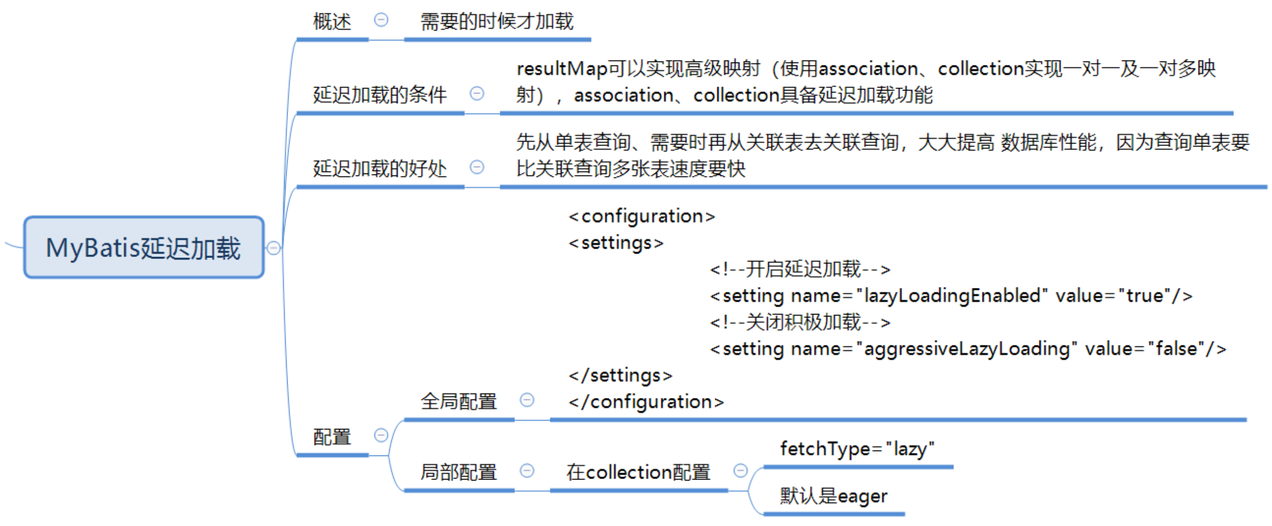
* 多对多

```

* 案例:
  * 查询亮哥信息及所销售商品的信息
  * 查询某个产品信息被谁销售过
* sql语句
-- 查询亮哥信息及所销售商品的信息
SELECT e.id eid,e.name ename,e.empno eempno,e.age eage,e.sex esex,e.job ejob,e.
g.id gid,g.goodsNo ggoodsNo,g.name gname,g.price gprice,g.num gnum
FROM employee e,goods g,eg eg_r WHERE e.empno=eg_r.eid AND g.goodsNo=eg_r.gid A

* Dao
public interface EmployeeDao {
    /**
     * @param name
     * @return
     * 查询亮哥信息及所销售商品的信息
     */
    Employee getEmployee(String name);
}
* 配置文件（记得在SqlMapConfig中注册）
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.lg.dao.EmployeeDao">
    <select id="getEmployee" parameterType="string" resultMap="rst">
        SELECT e.id eid,e.name ename,e.empno eempno,e.age eage,e.sex esex,e.jo
               g.id gid,g.goodsNo ggoodsNo,g.name gname,g.price gprice,g.num g
        FROM employee e,goods g,eg eg_r WHERE e.empno=eg_r.eid AND g.goodsNo=e
    </select>
    <resultMap id="rst" type="com.lg.bean.Employee">
        <result column="eid" property="id"></result>
        <result column="ename" property="name"></result>
        <result column="eempno" property="empno"></result>
        <result column="eage" property="age"></result>
        <result column="esex" property="sex"></result>
        <result column="ejob" property="job"></result>
        <collection property="goods" ofType="com.lg.bean.Goods">
            <result column="gid" property="id"></result>
            <result column="ggoodsNo" property="goodsNo"></result>
            <result column="gname" property="name"></result>
            <result column="gprice" property="price"></result>
            <result column="gnum" property="num"></result>
        </collection>
    </resultMap>
</mapper>
  @Test
public void test1(){
        SqlSession sqlSession = MyBatisUtils.getSqlSession();
        CompanyDao companyDao = sqlSession.getMapper(CompanyDao.class);
        Company company = companyDao.getCompany(1001);
        System.out.println(company);
        sqlSession.close();
 }

-- 查询某个产品信息被谁销售过
```

```
SELECT  e.id eid,e.name ename,e.empno eempno,e.age eage,e.sex esex,e.job ejob,e
g.id gid,g.goodsNo ggoodsNo,g.name gname,g.price gprice,g.num gnum
FROM employee e,goods g,eg eg_r WHERE e.empno=eg_r.eid AND g.goodsNo=eg_r.gid A

* Dao
public interface GoodsDao {
    /**
     * @param goodsNo
     * @return
     * 查询某个产品信息被谁销售过
     */
    Goods getGoods(int goodsNo);
}
* 配置文件
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.lg.dao.GoodsDao">
    <select id="getGoods" parameterType="int" resultMap="rst">
        SELECT  e.id eid,e.name ename,e.empno eempno,e.age eage,e.sex esex,e.jc
        g.id gid,g.goodsNo ggoodsNo,g.name gname,g.price gprice,g.num gnum
        FROM employee e,goods g,eg eg_r WHERE e.empno=eg_r.eid AND g.goodsNo=eg
    </select>

    <resultMap id="rst" type="com.lg.bean.Goods">
        <result column="gid" property="id"></result>
        <result column="ggoodsNo" property="goodsNo"></result>
        <result column="gname" property="name"></result>
        <result column="gprice" property="price"></result>
        <result column="gnum" property="num"></result>
        <collection property="employees" ofType="com.lg.bean.Employee">
            <result column="eid" property="id"></result>
            <result column="ename" property="name"></result>
            <result column="eempno" property="empno"></result>
            <result column="eage" property="age"></result>
            <result column="esex" property="sex"></result>
            <result column="ejob" property="job"></result>
        </collection>
    </resultMap>
```

```
</mapper>

* 单元测试
 @Test
public void test1(){
        SqlSession sqlSession = MyBatisUtils.getSqlSession();
        GoodsDao goodsDao = sqlSession.getMapper(GoodsDao.class);
        Goods goods = goodsDao.getGoods(10001);
        System.out.println(goods);
        sqlSession.close();
 }
```

* 能够掌握MyBatis延迟加载


MyBatis延迟加载

概述: 需要的时候才加载

延迟加载的条件: resultMap可以实现高级映射（使用association、collection实现一对一及一对多映射），association、collection具备延迟加载功能

延迟加载的好处: 先从单表查询、需要时再从关联表去关联查询，大大提高 数据库性能，因为查询单表要比关联查询多张表速度要快

配置:

全局配置:
```
<configuration>
<settings>
        <!--开启延迟加载-->
        <setting name="lazyLoadingEnabled" value="true"/>
        <!--关闭积极加载-->
        <setting name="aggressiveLazyLoading" value="false"/>
</settings>
</configuration>
```

局部配置: 在collection配置
- fetchType="lazy"
- 默认是eager

```
* 全局配置:
<configuration>
<settings>
        <!--开启延迟加载-->
        <setting name="lazyLoadingEnabled" value="true"/>
        <!--关闭积极加载-->
        <setting name="aggressiveLazyLoading" value="false"/>
</settings>
</configuration>
* 局部配置:
 <collection ... fetchType="lazy"/>

```

```
* 测试案例（SQL语句分开写的，才可以做延时加载）
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.lg.dao.CompanyDao">
    <select id="getCompany" parameterType="int" resultMap="rst">
        SELECT * FROM company c WHERE c.code=#{code}
    </select>
    <resultMap id="rst" type="com.lg.bean.Company">
        <collection property="departments" ofType="com.lg.bean.Department" colu
    </resultMap>
    <select id="getDepartments" parameterType="int" resultType="com.lg.bean.Dep
        SELECT * FROM department d WHERE d.cid=#{cid};
    </select>
</mapper>
public class CompanyDaoTest {
    @Test
    public void test1(){
        SqlSession sqlSession = MyBatisUtils.getSqlSession();
        CompanyDao companyDao = sqlSession.getMapper(CompanyDao.class);
        Company company = companyDao.getCompany(1001);
//          System.out.println(company); //打印compay已经去访问departments
        sqlSession.close();
    }
}
```