1、ResultMap(自定义数据库字段与pojo的映射关系），封装一个普通的pojo

```
<resultMap type="daiwei.test.pojo.User" id="myResultMap">
        <id column="id" property="id"/>
        <result column="name" property="theName"/>
        <result column="sex" property="sex"/>
        <result column="age" property="age"/>
    </resultMap>
```

type：要封装的pojo对象。

第一个id ： 为resultMap指定id供调用使用。

第二个id：被指定字段为数据库主键（会被一定的优化）。

result：指定数据库字段与pojo中属性相对应。

column：数据库指定要映射的字段；

property：pojo指定对应的属性；

2、ResultMap（级联属性封装）

```
<resultMap type="daiwei.test.pojo.User" id="complicateResultMap">
        <id column="id" property="id"/>
        <result column="name" property="theName"/>
        <result column="sex" property="sex"/>
        <result column="age" property="age"/>
        <result column="cid" property="city.cityId"/>
        <result column="cname" property="city.cityName"/>
    </resultMap>

<select id="getUserandCitybyId" resultMap="complicateResultMap">
        SELECT
            u.id id,
            u.`name` `name`,
            u.age age,
            u.sex sex,
            city_id cid,
```

```
            city_name cname
        FROM
            `user` u
        LEFT JOIN city c ON u.city_id = c.id
        WHERE
            u.id = #{id}
    </select>
```

封装对象User中包含一个City对象

封装city中cityId属性 通过 property="city.cityId"的方式封装

同理分装cityName属性通过property="city.cityName封装

3、ResultMap（使用association定义关联的单个对象的封装规则）

```
<resultMap type="daiwei.test.pojo.User" id="complicateResultMap1">
        <id column="id" property="id"/>
        <result column="name" property="theName"/>
        <result column="sex" property="sex"/>
        <result column="age" property="age"/>
        <association property="city" javaType="daiwei.test.pojo.City" >
            <id column="cid" property="cityId"/>
            <result column="cname" property="cityName"/>
        </association>
    </resultMap>
```

association 在一个resultMap中关联另外一个对象，在association标签中 property指定被关联对象中的属性名，javaType即为关联对象的全类名；

同样 association中的id为其关联对象的主键

result为关联对象的属性

4、ResultMap（使用association进行分步查询）

查询步骤：

1、先按照user的id属性查询User信息

2、根据User查询出来的city_id去city表中查询相关的City数据

3、把city属性封装到User中

```
<!-- public User getUserStepById(Integer id); -->
    <select id="getUserStepById" resultMap="complicateResultMap2">
        select * from user where id = #{id}
    </select>
     <resultMap type="daiwei.test.pojo.User" id="complicateResultMap2">
        <id column="id" property="id"/>
        <result column="name" property="theName"/>
        <result column="sex" property="sex"/>
        <result column="age" property="age"/>
        <association property="city"
            select="daiwei.test.myInterface.CityMapper.getCityById"
            column="city_id">
        </association>

    </resultMap>
```

以上查询的ResultMap association标签中

property：User中关联对象City的属性名

select：在city表中查询city有关信息的接口

column：要查询city表的传入的查询条件，在User表中的字段

以下是其他有关接口和配置文件的代码

1、cityMapper.xml

```
<resultMap type="daiwei.test.pojo.City" id="cityResultmap">
    <id column="id" property="cityId"/>
    <result column="city_name" property="cityName"/>
</resultMap>

<!-- public City getCityById(Integer cid); -->
    <select id="getCityById" resultMap="cityResultmap">
        select * from city where id = #{cid}
    </select>
</mapper>
```

2、cityMapper接口

```
public City getCityById(Integer cid);
```

思考：

对比以上两种方法这种方法的好处：sql分离，分步查询，sql语句可以写的很简单。

5、ResultMap associatioin延迟加载

关联的key只有在被使用的时候才会被加载，如果不被使用就不发送sql，减少数据库压力

只要在数据库中配置如下配置即可：

```
<settings>
        <setting name="lazyLoadingEnabled" value="true"/>
        <setting name="aggressiveLazyLoading" value="false"/>
    </settings>
```

6、ResultMap collection定义关联集合封装

在被封装的对象中包含一个集合时，通过在自定义的resultMap标签中添加collection标签实现

city中封装了个User的List；

```
public class City {

    private int cityId;
    private String cityName;
    private List<User> users;
}
```

mapper.xml中的封装方法：

```
<resultMap type="daiwei.test.pojo.City" id="cityResultMapPlus">
<!-- private int cityId;
    private String cityName;
    private List<User> users; -->
    <id column="id" property="cityId"/>
```

```
    <result column="cname" property="cityName"/>
    <collection property="users" ofType="daiwei.test.pojo.User">
    <!-- private int id;
    private String theName;
    private int age;
    private int sex;
    private City city; -->
        <id column="uid" property="id"/>
        <result column="name" property="theName"/>
        <result column="age" property="age"/>
        <result column="sex" property="sex"/>
    </collection>
</resultMap>
<!-- public City getCityByIdplus(Integer id); -->
<select id="getCityByIdplus" resultMap="cityResultMapPlus">
    SELECT
        c.id id,
        city_name cname,
        u.id uid,
        u. NAME NAME,
        u.age age,
        u.sex sex
    FROM
        city c
    LEFT JOIN `user` u ON u.city_id = c.id
    WHERE
        c.id = #{id};
</select>
```

7、ResultMap collection关联集合封装，进行分布查询延迟加载

cityMapper.xml中配置分步查询city中相关数据的第一个sql

```
<resultMap type="daiwei.test.pojo.City" id="cityResultMapPlus2">
    <id column="id" property="cityId"/>
    <result column="city_name" property="cityName"/>
    <collection property="users" ofType="daiwei.test.pojo.User"
    select="daiwei.test.myInterface.UserMapper1.getUserListBycid" column="id">
</collection>
</resultMap>
<select id="getCityStepByCId" resultMap="cityResultMapPlus2">
    select * from City where id = #{id}
```

```
</select>
```

以上查询的ResultMap collection标签中

property：City 中关联对象 User 集合的属性名

select：在`user`表中查询 city 有关信息的接口

column：传入的查询条件在`city`表中对应的相关字段，查询`user`表

userMapper.xml中配置完成分步查询用户有关信息的第二条sql

```
<resultMap type="daiwei.test.pojo.User" id="mySimpleResultMap">
    <id column="id" property="id"/>
    <result column="name" property="theName"/>
    <result column="sex" property="sex"/>
    <result column="age" property="age"/>
</resultMap>
<select id="getUserListBycid" resultMap="mySimpleResultMap">
    select * from user where city_id = #{cid}
</select>
```

如果要实现延迟加载按需加载的话也需要在mybatisConfig中开启相关的配置

```
<settings>
    <setting name="lazyLoadingEnabled" value="true"/>
    <setting name="aggressiveLazyLoading" value="false"/>
</settings>
```

关于 分步查询传参 和 延迟加载 的小拓展

1、如果分步查询第二次查询sql需要俩个参数，column如何传入多个参数：

——多列值通过map对传入参数进行封装，例如column="{ key1 = column1, key2 = column2}";

2、通过在 association 标签或 collection 标签中的 fetchType 属性来关闭 延迟加载

——默认情况 fetchType = “lazy” 即默认开始延迟加载，但是可以通过fetchType = “eager”来关闭延迟加载（即使在config配置文件中打开了延迟加载）

8、 discriminator鉴别器的使用（了解）

ResultMap 中的 discriminator 标签 通过查询出来的某个字段的不同情况来设置多个case 即多种不同的封装方案，以下是演示代码

```
<resultMap type="daiwei.test.pojo.User" id="complicateResultMap2">
    <id column="id" property="id"/>
    <result column="name" property="theName"/>
    <result column="sex" property="sex"/>
    <result column="age" property="age"/>
    <discriminator javaType="integer" column="sex">
        <case value="0" resultType="daiwei.test.pojo.User">
            <association property="city"
                select="daiwei.test.myInterface.CityMapper.getCityById"
                column="city_id">
            </association>
        </case>
        <case value="1" resultType="daiwei.test.pojo.User">
            <id column="id" property="id"/>
            <result column="name" property="theName"/>
            <result column="sex" property="sex"/>
            <result column="id" property="age"/>
        </case>
    </discriminator>

</resultMap>
```

discriminator 标签

javaTpye : 要鉴别的字段类型（mybatis别名）

case：value 控制的不同的情况

注：在这个例子中使用的是分步查询，如果 association 中，不使用select的方式 将获取不到city有关数据。