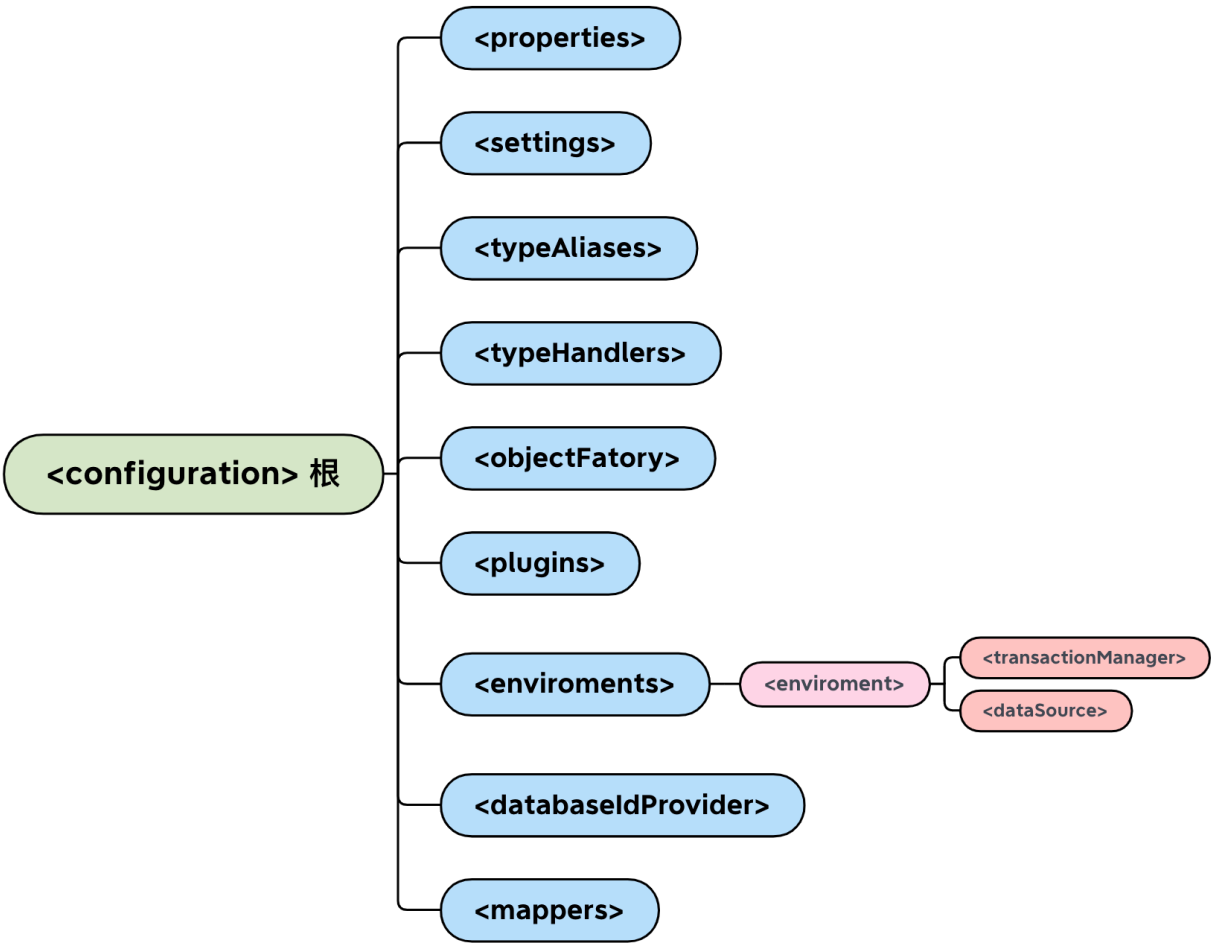
1 配置文件



1.1 <properties> 元素

> 用于将内部的配置外在化，例如用配置文件替换数据库的连接配置

- 即建立一个数据库 properties 文件，里面存储的是数据库的连接信息，在配置文件中通过 `${}` 获取

```
<properties resource = "db.properties">
```

1.2 <settiings> 元素

> 主要用于改变 MyBatis 运行时的行为，例如开启二级缓存、延迟加载

- `cacheEnabled` ： 所有映射器中配置的缓存全局开关，默认为false。
- `lazyLoadingEnabled` ： 延迟加载的全开关，默认为false。
- `aggressiveLazyLoading` ： 关联对象属性的延迟加载开关，默认为true（立即加载）。
- `multipleResultSetsEnabled` ： 是否允许单一语句返回多结果集 ，需兼容驱动，默认true。

- `useGeneratedKeys` ： 允许JDBC支持自动生成主键，需要驱动兼容，默认false。
- `autoMappingBehavior` ： 指定MyBatis如何自动映射列到字段或属性。

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

1.3 <typeAlias> 元素

```
为 java 类型设置一个别名
```

- 指定某个全限定类的别名

```
<typeAliases>
  <typeAliase alias = "newname" type = "com.x" />
</typeAliases>
```

- 自动扫描包，POJO小写作为别名

```
<typeAliases>
  <package name = "com.x"/>
</typeAliases>
```

1.4 <typeHandler> 元素

- 即将java 类型转换为对应数据库类型

```
用于将预处理语句中传入的参数从javaType（Java类型）转换为jdbcType(JDBC类型)，或者从数据库取出结果时将jdbcType转换为javaType。
```

- 当其内置的不能完成转换可以通过实现 `TypeHandler` 接口完成

```
<!-- 1 注册一个类的类型处理器 -->
<typeHandlers>
  <typeHandler handler = "com.bean.x"/>
</typeHandlers>


<!-- 2 注册一个包中所有类的类型处理器 -->
<typeHandlers>
  <package name = "com.bean"/>
</typeHandlers>
```

1.5 <enviroments> 元素

> 用于环境的配置，即数据源的配置，可以配置一种或多种

- 事务管理器
  - `JDBC`：提交和回滚，依赖于从数据源得到的连接管理事务作用域
  - `MANAGED`：不提交不回滚，容器管理事务的整个生命周期
- 数据源类型
  - `POOLED`：利用池，性能高
  - `UNPOOLED`：每次请求打开和关闭连接
  - `JNDI`：在 `EJB` 或应用服务器中使用

```
<!--配置环境，指定默认的数据库环境id为mysql-->
<environments default= "mysql">  <!-- 指定默认环境的 id -->
  <environment id= "mysql">    <!--配置 id 为 mysql 的数据库环境-->
    <transactionManager type= "JDBC" />  <!--使用JDBC的事务管理-->
    <dataSource type= "POOLED">          <!--数据库连接池-->
      < property name= "driver" value=""/>
      < property name= "url" value= "$[jdbc.url)"/>
      < property name= "username" value= "$[jdbc.username)"/>
      < property name= "password" value= "$(jdbc.password)"/>
    </dataSource>
  </environment>
< /environments>
```

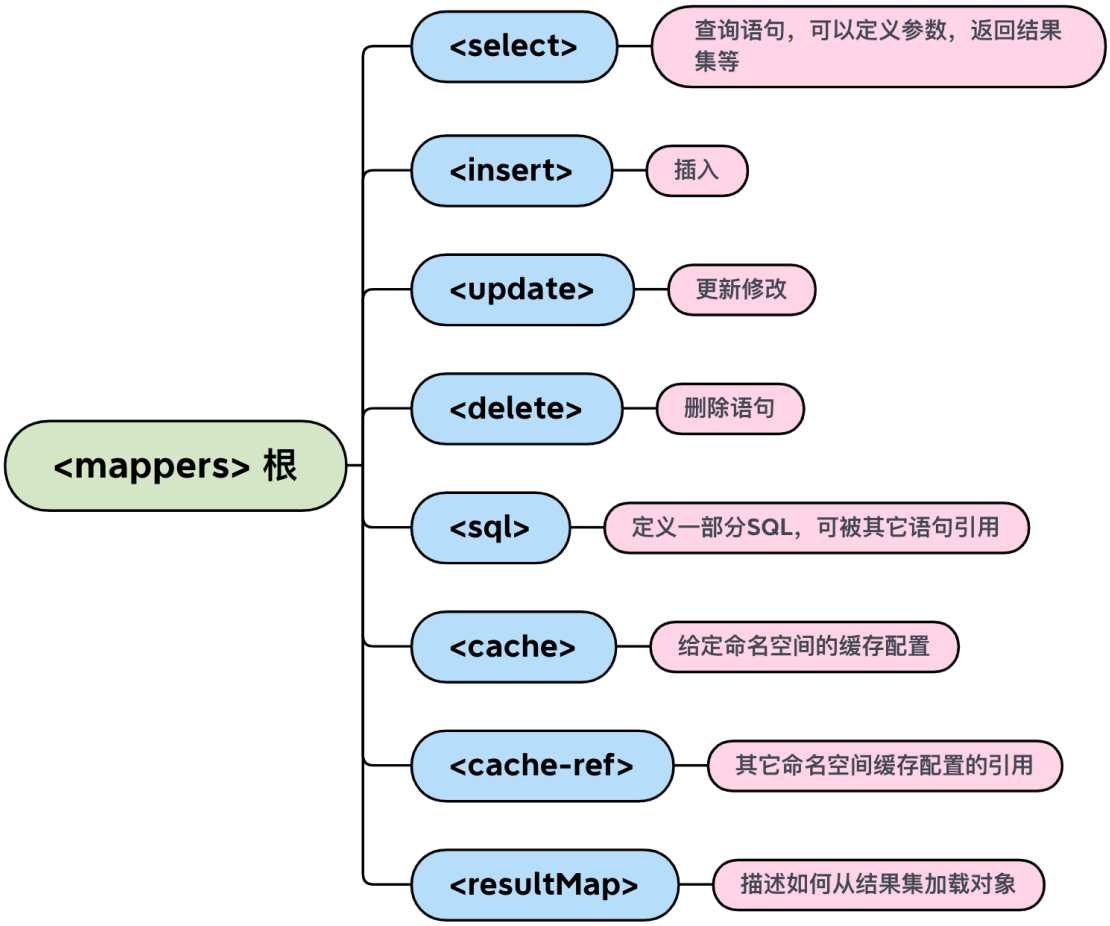
1.6 <mappers>

> 用于指定映射文件的位置

```
<!-- 1 类路径引入 -->
<mappers>
  <mapper resource = "resources/Mapper.xml" />
</mappers>

<!-- 2 包名引入（自动扫描包下所有 mapper 接口，需要创建映射文件对应的接口 -->
<mappers>
  <package name = "com.bean"/>
</mappers>

<!-- 3 接口类引入 -->
```

```
<mappers>
  <mapper class = "com.bean.UserMapper"/>
</mappers>

<!-- 4 本地文件路径引入 -->
<mappers>
  <mapper url = "file:///"/>
</mappers>
```

2 映射文件



`#{}`

- 占位符
- 可以防止 SQL 注入；
- 可以自动进行 Java 类型和 JDBC 类型的转换；
- 里面如果是字符类型或者是简单类型，可以写任意值；
- 里面如果是对象类型，解析符合 OGNL 表达式规则；

`${}`

- 用于模糊查询
- 是原封不动 SQL 拼接；
- 里面是字符类型或简单类型，只能写 value；
- 里面如果是对象类型，解析符合 OGNL 表达式规则；
- 无法防止 SQL 注入