# **What are the improvements that Mybatis is bigger than IBAtis?**

* There are interface bindings, including annotation binding sql and xml binding Sql,
* Dynamic sql is changed from the original node configuration to an OGNL expression.
* In the case of one-to-one, one-to-many, the association was introduced, and the collection node was introduced in one-to-many, but they were all configured in the resultMap.

# **What is the advantages of Mybait:**

* A number of third-party plugins (paged plugins / reverse engineering) are provided;
* Good integration with Spring;
* MyBatis is quite flexible and does not impose any influence on the existing design of the application or database. SQL is written in XML, completely separated from the program code, and the coupling between sql and program code is removed, which facilitates unified management and optimization. Reusable.
* Provide XML tags to support writing dynamic SQL statements.
* The writing of SQL statements is relatively large, especially when there are many fields and related tables, and this is especially true. There are certain requirements for developers to write SQL statements.
* The SQL statement depends on the database, resulting in poor portability of the database, and the database cannot be changed at will.
* MyBatis focuses on SQL itself and is a flexible enough DAO layer solution.
* MyBatis will be a good choice for projects with high performance requirements or more demand changes, such as Internet projects

# **What is the difference between MyBatis and Hibernate?**

1. Mybatis is different from hibernate. It is not an ORM framework. Because MyBatis requires programmers to write Sql statements themselves, but mybatis can flexibly configure the SQL statements to be run through XML or annotation, and map java objects and sql statements. Generate the final execution of sql, and finally map the results of the sql execution to generate a java object.
2. Hibernate object / relationship mapping ability, database independence, for the relationship model high software (such as fixed-demand custom software) if you use hibernate development can save a lot of code and improve efficiency. But Hibernate's shortcomings are high learning thresholds, higher proficiency thresholds,

# **What are the core processing classes of IBatis and MyBatis?**

The core processing class in IBatis is SqlMapClient. The core processing class in MyBatis is called SqlSession.

# **What are the differences in details between IBatis and MyBatis?**

(1) In sql, the variable name is changed to the original # variable# becomes #{variable} The original $variable $ becomes ${variable},

(2) The original class in the sql node is replaced by the name;

(3) The original queryForObject and queryForList become selectOne and selectList;

(4) The original alias setting is placed in the core configuration file in the mapping file;

# **What is the difference between #{} and ${}?**

#{} is precompiled, ${} is a string replacement.

When Mybatis processes #{}, it replaces #{} in sql with a ?, and calls the set method of PreparedStatement to assign a value;

When Mybatis processes ${}, it replaces ${} with the value of the variable.

Using #{} can effectively prevent SQL injection and improve system security.

What should I do if the attribute name in the entity class is different from the field name in the table?

Type 1: By defining the alias of the field name in the sql statement of the query, the alias of the field name is the same as the attribute name of the entity class.

    <select id=”selectorder” parametertype=”int” resultetype=”me.gacl.domain.order”>

select order\_id id, order\_no orderno ,order\_price price form orders where order\_id=#{id};

</select>

Type 2: Map the one-to-one correspondence between field names and entity class attribute names by <resultMap>

<select id="getOrder" parameterType="int" resultMap="orderresultmap">

select \* from orders where order\_id=#{id}

</select>

<resultMap type=”me.gacl.domain.order” id=”orderresultmap”>

<!–Map the primary key field with the id attribute –>

<id property=”id” column=”order\_id”>

<!–Use the result attribute to map non-primary key fields, property to entity class attribute names, and column to attributes in the data table –>

<result property = “orderno” column =”order\_no”/>

<result property=”price” column=”order\_price” />

</reslutMap>

How to write a fuzzy query like statement?

Type 1: Add sql wildcards to your Java code.

string wildcardname = “%smi%”;

list<name> names = mapper.selectlike(wildcardname);

<select id=”selectlike”>

select \* from foo where bar like #{value}

</select>

The second kind: splicing wildcards in sql statements will cause sql injection

string wildcardname = “smi”;

list<name> names = mapper.selectlike(wildcardname);

<select id=”selectlike”>

     select \* from foo where bar like "%"#{value}"%"

</select>

# **How is Mybatis paging? What is the principle of the paging plugin?**

Mybatis uses the RowBounds object for paging. It is a memory page for the ResultSet result set, not a physical page. You can directly write the parameters with physical paging in sql to complete the physical paging function, or you can use the paging plugin to complete the physical paging. .

The basic principle of the paging plugin is to use the plugin interface provided by Mybatis to implement the custom plugin, intercept the sql to be executed in the plugin interception method, and then rewrite the sql, according to the dialect dialect, add the corresponding physical paging statement and physical paging parameters.

# **How do I get the automatically generated (primary) key value?**

The insert method always returns an int value - this value represents the number of rows inserted.

The automatically generated key values ​​can be set to the passed in parameter object after the insert method has been executed.

Example:

<insert id=”insertname” usegeneratedkeys=”true” keyproperty=”id”>

insert into names (name) values (#{name})

</insert>

    name name = new name();

name.setname(“fred”);

int rows = mapper.insertname(name);

// After completion, the id has been set to the object

system.out.println(“rows inserted = ” + rows);

system.out.println(“generated key value = ” + name.getid());

# **How to pass multiple parameters in the mapper?**

<select id="selectUser"resultMap="BaseResultMap">

select \* fromuser\_user\_t whereuser\_name = #{0} anduser\_area=#{1}

</select

user selectuser(@param(“username”) string username,

@param(“hashedpassword”) string hashedpassword);

}

Then, you can use it in xml like this (recommended packaged as a map, passed as a single parameter to the mapper):

<select id=”selectuser” resulttype=”user”>

select id, username, hashedpassword

from some\_table

where username = #{username}

and hashedpassword = #{hashedpassword}

</select>

Another way to use map

Map<String, Object> map = new HashMap();

map.put("start", start);

map.put("end", end);

# **What does Mybatis Dynamic sql do? What dynamic sql? Can you briefly describe the implementation principle of dynamic sql?**

Mybatis dynamic sql allows us to write dynamic sql in the form of tags in the Xml mapping file, to complete the logic judgment and dynamic splicing sql function.

Mybatis provides 9 dynamic sql tags: trim|where|set|foreach|if|choose|when|otherwise|bind.

The execution principle is that OGNL is used to calculate the value of the expression from the sql parameter object, and the sql is dynamically spliced ​​according to the value of the expression, thereby completing the function of the dynamic sql.

# **In the Xml mapping file, in addition to the common select|insert|updae|delete tags, what other tags?**

A: There are many other tags, <resultMap>, <parameterMap>, <sql>, <include>, <selectKey>, plus 9 tags for dynamic sql, trim|where|set|foreach|if|choose| When|otherwise|bind, etc., where <sql> is the sql fragment tag, the sql fragment is introduced through the <include> tag, and <selectKey> generates the policy tag for the primary key that does not support auto-increment.

# **One-to-one, one-to-many association query?**

<mapper namespace="com.lcb.mapping.userMapper">

<!--association One-to-one association query -->

<select id="getClass" parameterType="int" resultMap="ClassesResultMap">

select \* from class c,teacher t where c.teacher\_id=t.t\_id and c.c\_id=#{id}

</select>

<resultMap type="com.lcb.user.Classes" id="ClassesResultMap">

<!-- Field name of entity class and field name mapping of data table -->

<id property="id" column="c\_id"/>

<result property="name" column="c\_name"/>

<association property="teacher" javaType="com.lcb.user.Teacher">

<id property="id" column="t\_id"/>

<result property="name" column="t\_name"/>

</association>

</resultMap>

<!--collection One-to-many association query -->

<select id="getClass2" parameterType="int" resultMap="ClassesResultMap2">

select \* from class c,teacher t,student s where c.teacher\_id=t.t\_id and c.c\_id=s.class\_id and c.c\_id=#{id}

</select>

<resultMap type="com.lcb.user.Classes" id="ClassesResultMap2">

<id property="id" column="c\_id"/>

<result property="name" column="c\_name"/>

<association property="teacher" javaType="com.lcb.user.Teacher">

<id property="id" column="t\_id"/>

<result property="name" column="t\_name"/>

</association>

<collection property="student" ofType="com.lcb.user.Student">

<id property="id" column="s\_id"/>

<result property="name" column="s\_name"/>

</collection>

</resultMap>

</mapper>

# **Mybatis's primary and secondary cache:**

1) Level 1 cache: PerpetualCache-based HashMap local cache, whose storage scope is Session. When Session flush or close, all Caches in the Session will be cleared, and the Level 1 cache is enabled by default.

2) The second level cache has the same mechanism as the level 1 cache. The default is also PerpetualCache, HashMap storage, except that its storage scope is Mapper (Namespace), and the storage source can be customized, such as Ehcache. By default, the second level cache is not enabled. To enable the level 2 cache, the level 2 cache attribute class needs to implement the Serializable serialization interface (which can be used to save the state of the object). You can configure <cache/> in its mapping file.

3) For the cache data update mechanism, when a C/U/D operation is performed on a scope (level cached session/secondary cache Namespaces), the cache in all selects under the scope will be cleared by default.

# **What is the interface binding of MyBatis, what are the benefits?**

Interface mapping is to define the interface arbitrarily in MyBatis, and then bind the method inside the interface with the SQL statement. We can directly call the interface method, so we can have more flexible options and settings than the original SqlSession method.

# **interface binding has several implementations, how are they implemented?**

There are two implementations of interface binding, one is through annotation binding, which is to add @Select, @Update and other annotations on the interface method, which contains Sql statement to bind; the other is to write SQL through xml To bind, in this case, you must specify the namespace in the xml mapping file must be the full path name of the interface.

# **What are the requirements for using the mapper interface of MyBatis?**

1 Mapper interface method name is the same as the id of each sql defined in mapper.xml  
2 The input parameter type of the Mapper interface method is the same as the type of each sql parameterType defined in mapper.xml  
3 The output parameter type of the Mapper interface method is the same as the type of each sql's resultType defined in mapper.xml.  
4 The namespace in the Mapper.xml file is the classpath of the mapper interface.