# MBATIS

# What is it?

- A JDBC Framework
- Developers write SQL, MyBATIS executes it using JDBC.
- An SQL Mapper
- Maps object properties to prepared statement parameters.
- Maps result sets to objects.

# What is not it(MyBATIS)?

- It is not an ORM
- Does not generate SQL
- Does not have a proprietary query language
- Does not build an object cache

# Point of Excellency

- MyBATIS  relys on your own objects, XML, and SQL.
- With the MyBATIS Data Mapper one have full power of both SQL and stored procedures at your fingertips.
- MyBATIS is a persistence framework that provides the benefits of SQL but avoids the complexity of JDBC. Unlike most other persistence frameworks, iBATIS encourages the direct use of SQL and ensures that all the benefits of SQL are not overridden by the framework itself.

# MyBATIS Design Features

- **Simplicity** − MyBatis is widely regarded as one of the simplest persistence frameworks available today.
- **Fast Development** − MyBatis does all it can to facilitate hyper-fast development.
- **Independent Interfaces** − MyBatis provides database-independent interfaces and APIs that help the rest of the application remain independent of any persistence-related resources.
- **Open source**− MyBatis is free and an open source software.

# Advantages of MYBATIS

- **Supports stored procedures** − MyBatis encapsulates SQL in the form of stored procedures so that business logic can be kept out of the database, and the application is more portable and easier to deploy and test.
- **Supports inline SQL** − No pre-compiler is needed, and you can have the full access to all of the features of SQL.
- **Supports dynamic SQL** − MyBatis provides features for dynamic building SQL queries based on parameters.

# MyBatis Installation

- To Download the latest version of MyBatis from Download MYBATIS.
- To Download the latest version of JDBC connector.
- To Unzip the downloaded files to extract .jar files and keep them in appropriate folders/directory.
- Setting CLASSPATH variable for the extracted .jar files appropriately.

# Necessary files for a simple example

- Configuration XML(.xml)
- Mapper XML(.xml)
- POJO Class(.java)
- Java class for opening and closing session (.java)

# Database Setup

- Let we create the fillowing table in database to run a simple example using MyBATIS

```
CREATE TABLE  student(
   ID int(10) NOT NULL IDENTITY(1,1),
   NAME varchar(100) NOT NULL,
   BRANCH varchar(255) NOT NULL,
   PERCENTAGE int(3) NOT NULL,
   PHONE int(10) NOT NULL,
   EMAIL varchar(255) NOT NULL,
   PRIMARY KEY ( ID )
);
```

# Configuration XML

- Since we are communicating with the database, we have to configure the details of the database. *Configuration XML* is the file used for the XML-based configuration. By using this file, you can configure various elements.

# Configuration XML(example)

```
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <environments default = "development">
    <environment id = "development">
      <transactionManager type = "JDBC"/>
      <dataSource type = "POOLED">
        <property name = "driver" value = "com.microsoft.sqlserver.jdbc.SQLServerDriver"/>
        <property name = "url" value = "jdbc:sqlserver://127.0.0.1;instance=MSSQLSERVER;DatabaseName=MyBATISdatabase"/>
        <property name = "username" value = "sa"/>
        <property name = "password" value = "0000"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource = "Student.xml"/>
  </mappers>
</configuration>
```

# environments tag

- Within the environments element, we configure the environment of the database that we use in our application. In MyBatis, one can connect to multiple databases by configuring multiple *environment* elements.
- To configure the environment, we are provided with two sub tags namely **transactionManager**and **dataSource**.

# transactionManager tag

- MyBatis supports two transaction managers namely **JDBC** and **MANAGED**
- **JDBC=** Responsible for the transaction management operations, such as, commit, roll-back, etc.
- **MANAGED**= the application server is responsible to manage the connection life cycle. Used in web applications.

# dataSource tag

- It is used to configure the connection properties of the database, such as driver-name, url, user-name, and password of the database that we want to connect. It is of three types namely
- **UNPOOLED=** For the dataSource type UNPOOLED, MyBatis simply opens and closes a connection for every database operation. Slower.
- **POOLED=** MyBatis will maintain a database connection pool. And, for every database operation, MyBatis uses one of these connections, and returns them to the pool after the completion of the operation. Faster.
- **JNDI** – For the dataSource type JNDI, MyBatis will get the connection from the JNDI dataSource.

# mappers tag

- Mapper’s element is used to configure the location of these mappers xml files in the configuration file of MyBatis
- The attribute **resource** points to the classpath of the XML file.

# Mapper XML(.xml)

- Mapper XML is an important file in MyBatis, which contains a set of statements to configure various SQL statements such as select, insert, update, and delete. These statements are known as **Mapped Statements** or **Mapped SQL Statements**.
- All the statements have unique id. To execute any of these statements you just need to pass the appropriate id to the methods in Java Application.(This is discussed clearly in later chapters).
- mapper XML file prevents the burden of writing SQL statements repeatedly in the application. In comparison to JDBC, almost 95% of the code is reduced using Mapper XML file in MyBatis.
- All these Mapped SQL statements are resided within the element named**<mapper>**. This element contains an attribute called**‘namespace’**.

# Mapper XML(example)

```
<?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 = "Student">
   <resultMap id = "result" type = "Student">
      <result property = "id" column = "ID"/>
      <result property = "name" column = "NAME"/>
      <result property = "branch" column = "BRANCH"/>
      <result property = "percentage" column = "PERCENTAGE"/>
      <result property = "phone" column = "PHONE"/>
      <result property = "email" column = "EMAIL"/>
   </resultMap>
   <select id = "getAll" resultMap = "result">
      SELECT * FROM STUDENT;
   </select>
   <select id = "getById" parameterType = "int" resultMap = "result">
      SELECT * FROM STUDENT WHERE ID = #{id};
   </select>
</mapper>
```

# POJO Class(.java)

- To perform any Create, Read, Update, and Delete (CRUD) operation using MyBATIS, you would need to create a Plain Old Java Objects (POJO) class corresponding to the table. This class describes the objects that will "model" database table rows.
- The POJO class would have implementation for all the methods required to perform desired operations.

# POJO Class(example)

```
public class Student {
   private int id;
   private String name;
   private String branch;
   private int percentage;
   private int phone;
   private String email;
   public Student(int id, String name, String branch, int percentage, int phone, String email) {
      super();
      this.id = id;
      this.name = name;
      this.branch = branch;
      this.percentage = percentage;
      this.phone = phone;
      this.email = email;
```

```
}
  public Student() {}
  public int getId() {
    return id;
  }
  public String getName() {
    return name;
  }
  public int getPhone() {
    return phone;
  }
  public String getEmail() {
    return email;
  }
```

```
 public String getBranch() {
     return branch;
   }
   public int getPercentage() {
     return percentage;
   }
}
```

# Application level Java class (.java)

- This file would have application level logic to insert records in the Student table.

# Application level Java class (example)

```
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;
public class mybatisRead_ALL {
   public static void main(String args[]) throws IOException{
      Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
      SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);
      SqlSession session = sqlSessionFactory.openSession();
```

```
 //select contact all contacts
     List<Student> student = session.selectList("Student.getAll");
     for(Student st : student ){
       System.out.println(st.getId());
       System.out.println(st.getName());
       System.out.println(st.getBranch());
       System.out.println(st.getPercentage());
       System.out.println(st.getEmail());
       System.out.println(st.getPhone());
     }
     System.out.println("Records Read Successfully ");
     session.commit();
     session.close();
   }
}
```

# Annotation

- In the previous chapters, we have seen how to perform curd operations using MyBatis. There we used a Mapper XML file to store mapped SQL statements and a configuration XML file to configure MyBatis.
- To map SQL statements, MyBatis also provides annotations. So, this chapter discusses how to use MyBatis annotations.
- While working with annotations, instead of configuration XML file, we can use a java mapper interface to map and execute SQL queries.

# Annotation(example)

```
import java.util.List;
import org.apache.ibatis.annotations.*;
public interface Student_mapper {
  final String getAll = "SELECT * FROM STUDENT";
  @Select(getAll)
  @Results(value = {
    @Result(property = "id", column = "ID"),
    @Result(property = "name", column = "NAME"),
    @Result(property = "branch", column = "BRANCH"),
    @Result(property = "percentage", column = "PERCENTAGE"),
    @Result(property = "phone", column = "PHONE"),
    @Result(property = "email", column = "EMAIL")
  })
  List getAll();
}
```

```
public class Annotations_Example {
   public static void main(String args[]) throws IOException{
      Reader reader =
Resources.getResourceAsReader("mybatisAnnotation\\SqlMapConfig.xml");
      SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(reader);
      SqlSession session = sqlSessionFactory.openSession();
      session.getConfiguration().addMapper(Student_mapper.class);
      Student_mapper mapper = session.getMapper(Student_mapper.class);
mapper.select();
```

# MyBatis vs Hibernate

- Both Hibernate and MyBatis are open source Object Relational Mapping (ORM) tools available in the industry. Use of each of these tools depends on the context you are using them.

# MyBatis vs Hibernate(contd.)

| MyBatis | Hibernate |
| --- | --- |
| It is simpler. It comes in a much smaller package size. | Hibernate generates SQL for you, which means you don’t have to spend time on generating SQL. |
| It is flexible, offers faster development time. | It is highly scalable, provides a much more advanced cache. |
| It uses SQL, which could be database dependent. | It uses HQL, which is relatively independent of databases. It is easier to change db into Hibernate. |
| It maps the ResultSet from JDBC API to your POJO Objects, so you don’t have to care about table structures. | Hibernate maps your Java POJO objects to the Database tables. |
| It is quite easy to use stored procedure in MyBatis. | Use of stored procedures is a little difficult in Hibernate. |

# MyBatis is suggested in following cases:

- You want to create your own SQL's and you are willing to maintain them.
- Your environment is driven by relational data model.
- You have to work on existing and complex schemas.

# Literature


- https://en.wikipedia.org/wiki/MyBatis
- http://respect-architects.blogspot.ru/2013/08/sql-data-management-with-mybatis.html
- http://stackoverflow.com/questions/1984548/hibernate-vs-ibatis
- https://en.wikipedia.org/wiki/IBATIS
- http://www.javaworld.com/article/2077875/open-source-tools/ibatis--hibernate--and-jpa--which-is-right-for-you-.html
- http://stackoverflow.com/questions/2867325/migrating-from-hand-written-persistence-layer-to-orm

# That is all!!

Thanks for attention