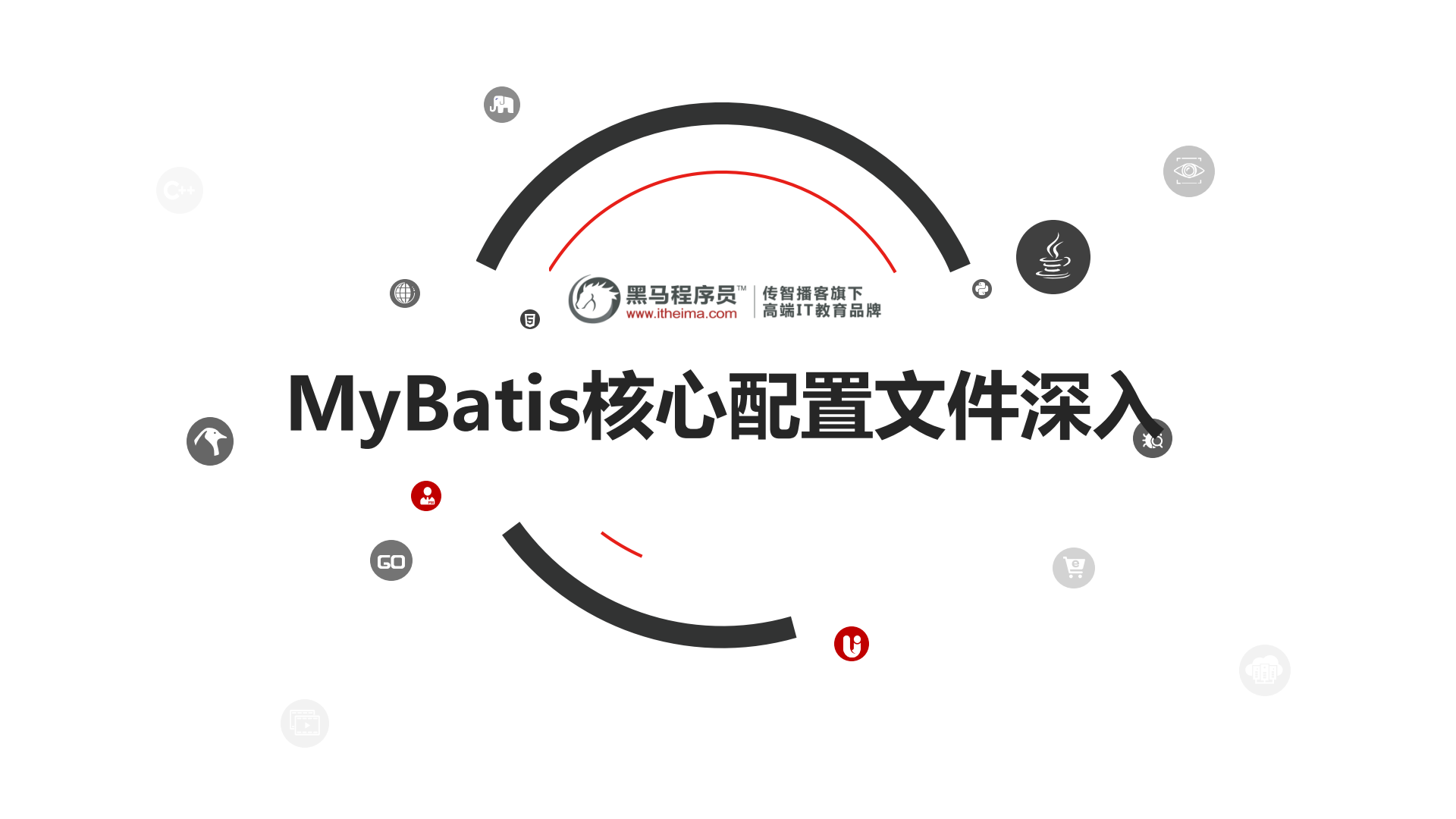

MyBatis核心配置文件深入

目录 Contents

1. MyBatis核心配置文件深入

1.1 typeHandlers标签

无论是 MyBatis 在预处理语句（PreparedStatement）中设置一个参数时，还是从结果集中取出一个值时， 都会用类型处理器将获取的值以合适的方式转换成 Java 类型。下表描述了一些默认的类型处理器（截取部分）。

类型处理器	Java 类型	JDBC 类型
`BooleanTypeHandler`	`java.lang.Boolean`, `boolean`	数据库兼容的 `BOOLEAN`
`ByteTypeHandler`	`java.lang.Byte`, `byte`	数据库兼容的 `NUMERIC` 或 `BYTE`
`ShortTypeHandler`	`java.lang.Short`, `short`	数据库兼容的 `NUMERIC` 或 `SHORT INTEGER`
`IntegerTypeHandler`	`java.lang.Integer`, `int`	数据库兼容的 `NUMERIC` 或 `INTEGER`
`LongTypeHandler`	`java.lang.Long`, `long`	数据库兼容的 `NUMERIC` 或 `LONG INTEGER`

1. MyBatis核心配置文件深入

1.1 typeHandlers标签

你可以重写类型处理器或创建你自己的类型处理器来处理不支持的或非标准的类型。具体做法为：实现 org.apache.ibatis.type.TypeHandler 接口， 或继承一个很便利的类 org.apache.ibatis.type.BaseTypeHandler， 然后可以选择性地将它映射到一个JDBC类型。例如需求：一个Java中的Date数据类型，我想将之存到数据库的时候存成一个1970年至今的毫秒数，取出来时转换成java的Date，即java的Date与数据库的varchar毫秒值之间转换。

开发步骤：

① 定义转换类继承类BaseTypeHandler<T>

② 覆盖4个未实现的方法，其中setNonNullParameter为java程序设置数据到数据库的回调方法，getNullableResult为查询时 mysql的字符串类型转换成 java的Type类型的方法

③ 在MyBatis核心配置文件中进行注册

④ 测试转换是否正确

1. MyBatis核心配置文件深入

1.1 typeHandlers标签

```
public class MyDateTypeHandler extends BaseTypeHandler<Date> {
    public void setNonNullParameter(PreparedStatement preparedStatement, int i, Date date, JdbcType type)
{
        preparedStatement.setString(i,date.getTime()+"");
    }
    public Date getNullableResult(ResultSet resultSet, String s) throws SQLException {
        return new Date(resultSet.getLong(s));
    }
    public Date getNullableResult(ResultSet resultSet, int i) throws SQLException {
        return new Date(resultSet.getLong(i));
    }
    public Date getNullableResult(CallableStatement callableStatement, int i) throws SQLException {
        return callableStatement.getDate(i);
    }
}
```

1. MyBatis核心配置文件深入

1.1 typeHandlers标签

```
<!--注册类型自定义转换器-->
<typeHandlers>
    <typeHandler handler="com.itheima.typeHandlers.MyDateTypeHandler"></typeHandler>
</typeHandlers>
```

测试添加操作：

```
user.setBirthday(new Date());
userMapper.add2(user);
```

数据库数据：

id	username	password	birthday
1	lucy	123	1539751863457
2	tom	123	1539751863457
5	haohao	123	1539751863457

1. MyBatis核心配置文件深入

1.1 typeHandlers标签

测试查询操作：

```
13:53:10,222 DEBUG findAll:159 - <==      Total: 9
[User{id=1, username='lucy', password='123', birthday=Wed Oct 17 12:51:03 GMT+08:00 2018}
13:53:10,222 DEBUG JdbcTransaction:123 - Resetting autocommit to true on JDBC Connection
13:53:10,222 DEBUG JdbcTransaction:91 - Closing JDBC Connection [com.mysql.jdbc.JDBC4Conne
13:53:10,222 DEBUG PooledDataSource:363 - Returned connection 249155636 to pool.
```

1. MyBatis核心配置文件深入

1.2 plugins标签

MyBatis可以使用第三方的插件来对功能进行扩展，分页助手PageHelper是将分页的复杂操作进行封装，使用简单的方式即可获得分页的相关数据

开发步骤：

① 导入通用PageHelper的坐标

② 在mybatis核心配置文件中配置PageHelper插件

③ 测试分页数据获取

1. MyBatis核心配置文件深入

1.2 plugins标签

① 导入通用PageHelper坐标

```
<!-- 分页助手 -->
<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
    <version>3.7.5</version>
</dependency>
<dependency>
    <groupId>com.github.jsqlparser</groupId>
    <artifactId>jsqlparser</artifactId>
    <version>0.9.1</version>
</dependency>
```

1. MyBatis核心配置文件深入

1.2 plugins标签

② 在mybatis核心配置文件中配置PageHelper插件

```
<!-- 注意：分页助手的插件  配置在通用馆mapper之前 -->
<plugin interceptor="com.github.pagehelper.PageInterceptor">


</plugin>
```

原版本

```
<!--    <plugins>-->
<!--        <plugin interceptor="com.github.pagehelper.PageHelper">-->
<!--            <property name="dialect" value="mysql"/>-->
<!--        </plugin>-->
<!--    </plugins>-->
```

1. MyBatis核心配置文件深入

1.2 plugins标签

③ 测试分页代码实现

```
@Test
public void testPageHelper(){
   //设置分页参数
       PageHelper.startPage(1,2);

   List<User> select = userMapper2.select(null);
   for(User user : select){
      System.out.println(user);
   }
}
```

1. MyBatis核心配置文件深入

1.2 plugins标签

获得分页相关的其他参数

```
//其他分页的数据
PageInfo<User> pageInfo = new PageInfo<User>(select);
System.out.println("总条数： "+pageInfo.getTotal());
System.out.println("总页数： "+pageInfo.getPages());
System.out.println("当前页： "+pageInfo.getPageNum());
System.out.println("每页显示长度： "+pageInfo.getPageSize());
System.out.println("是否第一页： "+pageInfo.isIsFirstPage());
System.out.println("是否最后一页： "+pageInfo.isIsLastPage());
```

1. MyBatis核心配置文件深入

1.3 知识小结

MyBatis核心配置文件常用标签：

1、properties标签：该标签可以加载外部的properties文件

2、typeAliases标签：设置类型别名

3、environments标签：数据源环境配置标签

4、typeHandlers标签：配置自定义类型处理器

5、plugins标签：配置MyBatis的插件

```xml
<configuration>
    <properties resource="jdbc.properties"></properties>

    <typeAliases>
        <typeAlias type="example.domain.User" alias="user"></typeAlias>
    </typeAliases>

<!--    自定义注册类型转换器-->
    <typeHandlers>
        <typeHandler handler="example.handler.DateTypeHandler"></typeHandler>
    </typeHandlers>

<!--    配置分页助手插件-->
    <plugins>
        <plugin interceptor="com.github.pagehelper.PageInterceptor"></plugin>
    </plugins>


    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC"></transactionManager>
            <dataSource type="POOLED">
                <property name="driver" value="${jdbc.driver}"/>
                <property name="url" value="${jdbc.url}"/>
                <property name="username" value="${jdbc.user}"/>
                <property name="password" value="${jdbc.password}"/>
            </dataSource>
        </environment>
    </environments>

    <mappers>
        <mapper resource="UserMapper.xml"></mapper>
    </mappers>

</configuration>
```

传智播客旗下高端IT教育品牌