

## [今日课程大纲]

<settings>标签常用属性介绍

MyBatis 对 Log4J 的支持

Log4J 复习

parameterType 属性(对象,map,基本类型,#{},\${}等)

<typeAliases>两种别名配置及内置别名

MyBatis 实现新增及 MyBatis 事务

MyBatis 实现删除

MyBatis 实现修改

## [知识点详解]

### 一.注解

1.注解存在的意义:简化 xml 文件的开发.

2.注解在 servlet 3.0 规范之后大力推广的.

3.注解前面的@XXX,表示引用一个@interface

3.1 @interface 表示注解声明

4.注解可以有属性,因为注解其实就是一个接口(类)

4.1 每次使用注解都需要导包

5.注解语法: @XXXX(属性名= 值)

6.值的分类

6.1 如果值是基本数据类型或字符串: 属性名=值

6.2 如果值是数组类型: 属性名={值,值}

6.2.1 如果只有一个值可以省略大括号

6.3 如果值是类类型,属性名=@名称

7.如果注解只需要给一个属性赋值,且这个属性是默认属性,可以省略属性名

## 二. 路径

1. 编写路径为了告诉编译器如何找到其他资源.

2. 路径分类

2.1 相对路径: 从当前资源出发找到其他资源的过程

2.2 绝对路径: 从根目录(服务器根目录或项目根目录)出发找到其他资源的过程

2.2.1 标志: 只要以/开头的都是绝对路径

3. 绝对路径:

3.1 如果是请求转发 / 表示项目根目录(WebContent)

3.2 其他重定向,<img/> <script/>,<style/>,location.href 等/都表示服务器根目录(tomcat/webapps 文件夹)

4. 如果客户端请求的控制器,控制器转发到JSP后,jsp中如果使用相对路径,需要按照控制器的路径去找其他资源.

4.1 保险办法:使用绝对路径,可以防止上面的问题.

## 三. Log4J

1. 由 apache 推出的开源免费日志处理的类库.

2. 为什么需要日志:

2.1 在项目中编写 `System.out.println();` 输出到控制台,当项目发布到 tomcat 后,没有控制台(在命令行界面能看见.),不容易观察一些输出结果.

2.2 log4j 作用,不仅能把内容输出到控制台,还能把内容输出到文件中.便于观察结果.

3. 使用步骤:

3.1 导入 log4j-xxx.jar

3.2 在 src 下新建 log4j.properties(路径和名称都不允许改变)

3.2.1 ConversionPattern :写表达式

3.2.2 log4j.appender.LOGFILE.File 文件位置及名称(日志文件扩展名.log)

```
log4j.rootCategory=DEBUG, CONSOLE ,LOGFILE

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender

log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout

log4j.appender.CONSOLE.layout.ConversionPattern=%C %d{
```

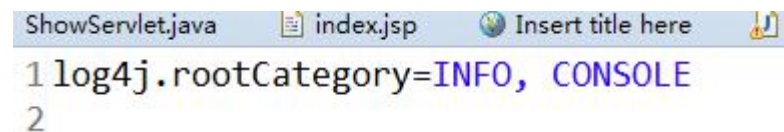
```
YYYY-MM-dd hh:mm:ss} %m %n

log4j.appender.LOGFILE=org.apache.log4j.FileAppender
log4j.appender.LOGFILE.File=E:/my.log
log4j.appender.LOGFILE.Append=true
log4j.appender.LOGFILE.layout=org.apache.log4j.Pattern
Layout
log4j.appender.LOGFILE.layout.ConversionPattern=%C %m
%L %n
```

#### 4. log4j 输出级别

4.1 fatal(致命错误) > error (错误) > warn (警告) > info(普通信息) > debug(调试信息)

4.2 在 log4j.properties 的第一行中控制输出级别

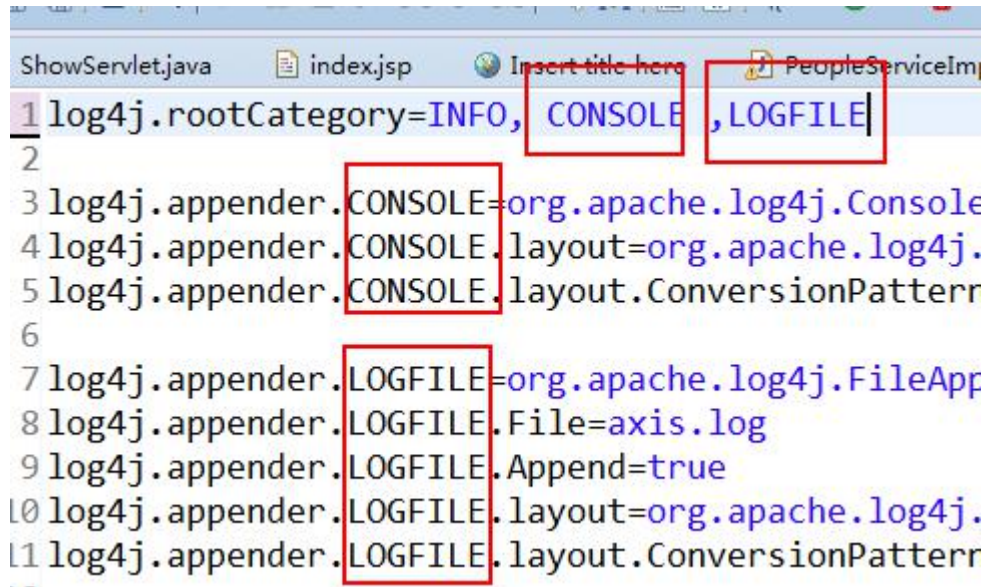


The screenshot shows a web browser with three tabs: 'ShowServlet.java', 'index.jsp', and 'Insert title here'. The 'index.jsp' tab is active, displaying the following code:

```
1 log4j.rootCategory=INFO, CONSOLE
2
```

#### 5. log4j 输出目的地

5.1 在一行控制输出目的地



```

1 log4j.rootCategory=INFO, CONSOLE, LOGFILE
2
3 log4j.appender.CONSOLE=org.apache.log4j.Console
4 log4j.appender.CONSOLE.layout=org.apache.log4j.
5 log4j.appender.CONSOLE.layout.ConversionPattern
6
7 log4j.appender.LOGFILE=org.apache.log4j.FileApp
8 log4j.appender.LOGFILE.File=axis.log
9 log4j.appender.LOGFILE.Append=true
10 log4j.appender.LOGFILE.layout=org.apache.log4j.
11 log4j.appender.LOGFILE.layout.ConversionPattern

```

## 6. pattern 中常用几个表达式

6.1 %C 包名+类名

6.2 %d{YYYY-MM-dd HH:mm:ss} 时间

6.3 %L 行号

6.4 %m 信息

6.5 %n 换行

## 四. <settings>标签

1. 在 mybatis 全局配置文件中通过<settings>标签控制 mybatis 全局开关

2. 在 mybatis.xml 中开启 log4j

2.1 必须保证有 log4j.jar

2.2 在 src 下有 log4j.properties

<settings>

```
<setting name="LogImpl" value="LOG4J"/>

</settings>
```

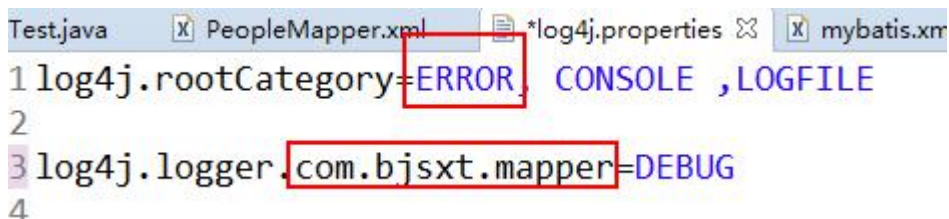
3.log4j 中可以输出指定内容的日志(控制某个局部内容的日志级别)

3.1 命名级别(包级别): <mapper>namespace 属性中除了最后一个类名

例如 namespace="com.bjsxt.mapper.PeopleMapper" 其中包级别为 com.bjsxt.mapper ,需要在 log4j.properties 中

3.1.1 先在总体级别调成 Error 不输出无用信息

3.1.2 在设置某个指定位置级别为 DEBUG



The screenshot shows the log4j.properties file with the following content:

```
1 log4j.rootCategory=ERROR, CONSOLE, LOGFILE
2
3 log4j.logger.com.bjsxt.mapper=DEBUG
4
```

Red boxes highlight the 'ERROR' value in line 1 and the 'com.bjsxt.mapper' package name in line 3.

3.2 类级别

3.2.1 namespace 属性值 ,namespace 类名

3.3 方法级别

3.3.2 使用 namespace 属性值+标签 id 属性值

## 五 parameterType 属性

1. 在 XXXMapper.xml 中<select><delete>等标签的 parameterType 可以控制参数类型

2. SqlSession 的 selectList()和 selectOne()的第二个参数和 selectMap()的第三个参数都表示方法的参数.

## 2.1 示例,

```
People p = session.selectOne("a.b.selById",1);  
System.out.println(p);
```

## 2.2 在 Mapper.xml 中可以通过#{ }获取参数

### 2.2.1 parameterType 控制参数类型

### 2.2.2 #{ }获取参数内容

#### 2.2.2.1 使用索引,从 0 开始 #{0}表示第一个参数

#### 2.2.2.2 也可以使用#{param1}第一个参数

#### 2.2.2.3 如果只有一个参数(基本数据类型或 String),mybatis

对#{ }里面内容没有要求只要写内容即可.

#### 2.2.2.4 如果参数是对象#{属性名}

#### 2.2.2.5 如果参数是 map 写成#{key}

```
<select id="selById"  
resultType="com.bjsxt.pojo.People"  
parameterType="int">  
    select * from people where id=#{0}  
</select>
```

## 3. #{ } 和 \${ } 的区别

3.1 #{ } 获取参数的内容支持 索引获取,param1 获取指定位置参数,  
并且 SQL 使用?占位符

3.2 \${ } 字符串拼接不使用?,默认找\${内容}内容的 get/set 方法,如  
果写数字,就是一个数字

#### 4. 如果在 xml 文件中出现 “<”, “>”, 双引号 等特殊字符时可以使用 XML 文件转义标签(XML 自身的)

4.1 <![CDATA[ 内容 ]]>

#### 5. mybatis 中实现 mysql 分页写法

5.1 ?不允许在关键字前后进行数学运算,需要在代码中计算完成后传递到 mapper.xml 中

5.2 在 java 代码中计算

```
//显示几个
int pageSize = 2;

//第几页
int pageNumber = 2;

//如果希望传递多个参数,可以使用对象或 map
Map<String,Object> map = new HashMap<>();
map.put("pageSize", pageSize);
map.put("pageStart", pageSize*(pageNumber-1));
List<People> p =
session.selectList("a.b.page",map);
```

5.3 在 mapper.xml 中代码

```
<select id="page" resultType="com.bjsxt.pojo.People"
parameterType="map">
    select * from people limit #{pageStart},#{pageSize}
</select>
```



## 六. typeAliases 别名

1.系统内置别名: 把类型全小写

2.给某个类起别名

2.1 alias="自定义"

```
<typeAliases>
    <typeAlias type="com.bjsxt.pojo.People"
alias="peo"/>
</typeAliases>
```

2.2 mapper.xml 中 peo 引用 People 类

```
<select id="page" resultType="peo"
parameterType="map">
    select * from people limit #{pageStart},#{pageSize}
</select>
```

3.直接给某个包下所有类起别名,别名为类名,不区分大小写

3.1 mybatis.xml 中配置

```
<typeAliases>
    <package name="com.bjsxt.pojo" />
</typeAliases>
```

3.2 mapper.xml 中通过类名引用

```
<select id="page" resultType="People"
parameterType="map">
    select * from people limit #{pageStart},#{pageSize}
</select>
```

## 七.MyBatis 实现新增

### 1. 概念复习

1.1 功能:从应用程序角度出发,软件具有哪些功能.

1.2 业务:完成功能时的逻辑.对应 Service 中一个方法

1.3 事务:从数据库角度出发,完成业务时需要执行的 SQL 集合,统称一个事务.

1.3.1 事务回滚.如果在一个事务中某个 SQL 执行事务,希望回归到事务的原点,保证数据库数据的完整性.

### 2. 在 mybatis 中默认是关闭了 JDBC 的自动提交功能

2.1 每一个 SqlSession 默认都是不自动提交事务.

2.2 session.commit()提交事务.

2.3 openSession(true);自动提交.setAutoCommit(true);

### 3. mybatis 底层是对 JDBC 的封装.

3.1 JDBC 中 executeUpdate()执行新增,删除,修改的 SQL.返回值 int,表示受影响的行数.

3.2 mybatis 中<insert> <delete> <update>标签没有 resultType 属性,认为返回值都是 int

4. 在 `openSession()` 时 Mybatis 会创建 `SqlSession` 时同时创建一个 `Transaction`(事务对象),同时 `autoCommit` 都为 `false`

4.1 如果出现异常,应该 `session.rollback()`回滚事务.

5. 实现新增的步骤

5.1 在 `mapper.xml` 中提供`<insert>`标签,标签没有返回值类型

```
<insert id="ins" parameterType="People">
    insert into people values(default,#{name},#{age})
</insert>
```

5.2 通过 `session.insert()`调用新增方法

```
int index1 = session.insert("a.b.ins", p);
if(index1>0){
    System.out.println("成功");
}else{
    System.out.println("失败");
}
```

## 八.MyBatis 实现修改

1. 在 `mapper.xml` 中提供`<update>`标签

```
<update id="upd" parameterType="People">
    update people set name = #{name} where id = #{id}
</update>
```

2. 编写代码

```
People peo = new People();  
peo.setId(3);  
peo.setName("王五");  
int index = session.update("a.b.upd", peo);  
if(index>0){  
    System.out.println("成功");  
}else{  
    System.out.println("失败");  
}  
  
session.commit();
```

## 九.mybatis 实现删除

1. 在 mapper.xml 提供<delete>标签

```
<delete id="del" parameterType="int">  
    delete from people where id = #{0}  
</delete>
```

2. 编写代码

```
int del = session.delete("a.b.del",3);  
  
if(del>0){  
    System.out.println("成功");  
}
```

```
}else{  
    System.out.println("失败");  
}  
  
session.commit();
```