

Mybatis_day1

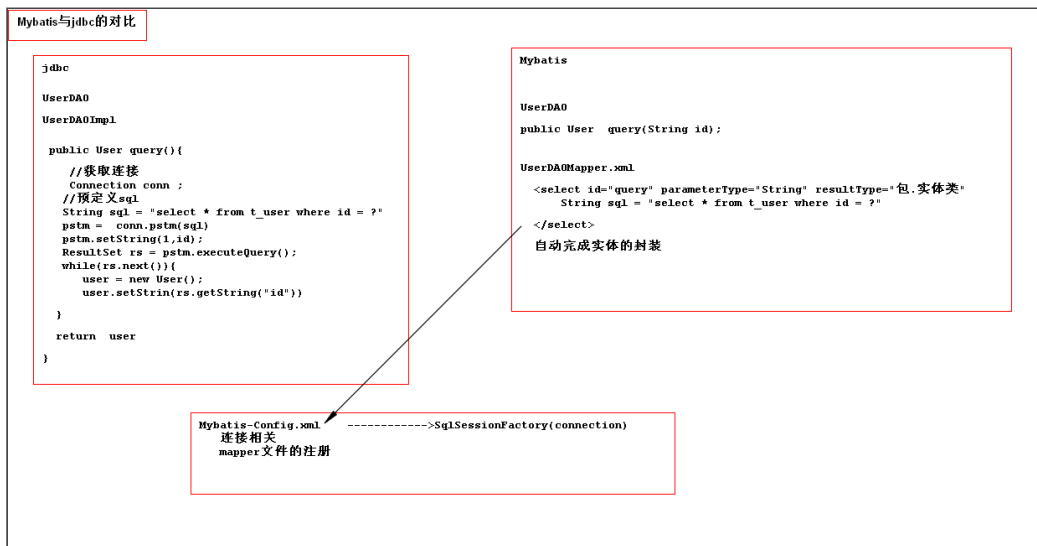
1. Mybatis的引言

Mybatis是一个基于java语言的持久层的框架，主要用来简化数据库的访问操作,内部封装了原来的jdbc代码,替换了原有项目开发中的jdbc技术,他可以自动完成对象与关系的映射(OR),极大简化了我们的开发,提高开发效率。

2. Jdbc中存在问题

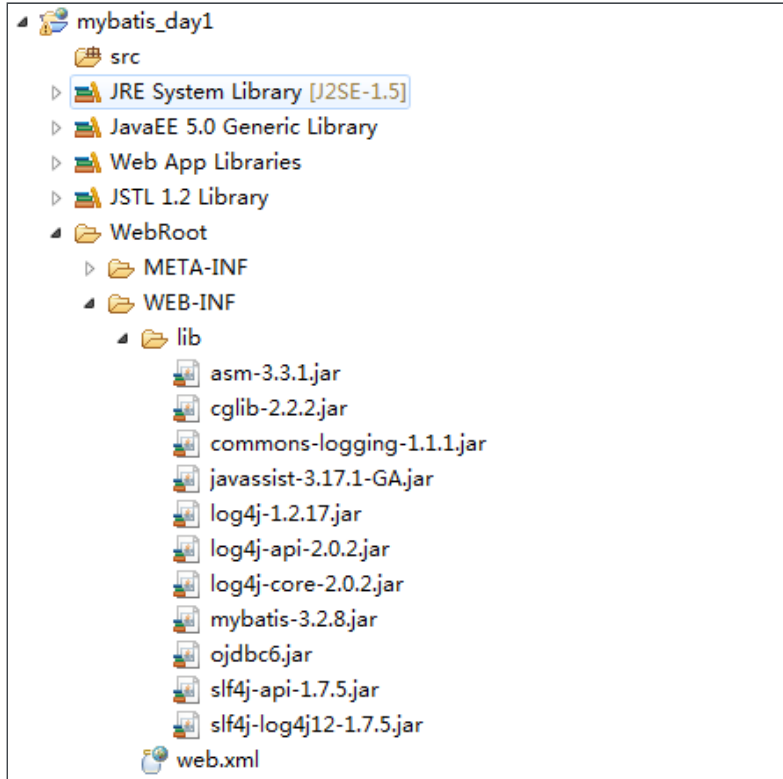
- 1).大量的代码冗余 (处理结果集的时候存在大量的冗余)
- 2).不能完成数据库和实体的自动转换 (需要手动封装实体,不能自定封装实体类)

3. Mybatis框架的开发思路



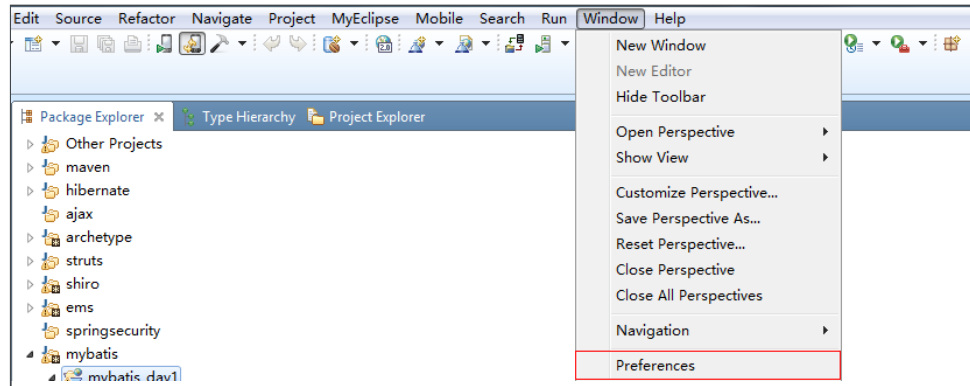
4. Mybatis的环境搭建

4.1. 引入jar包(mybatis核心以及驱动)

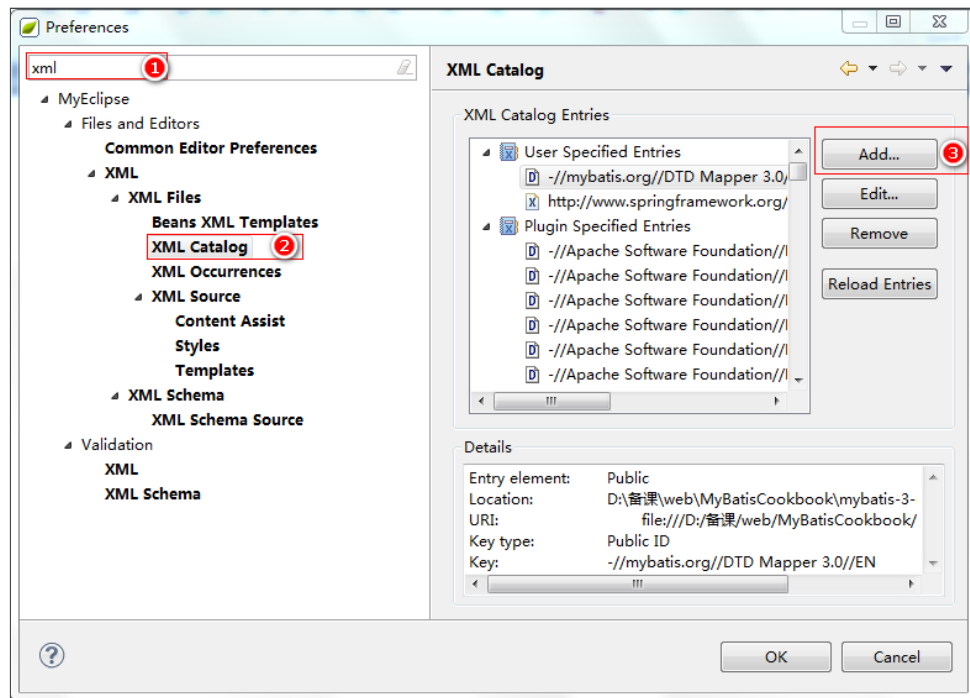


4.2. 配置mybatis的提示文件

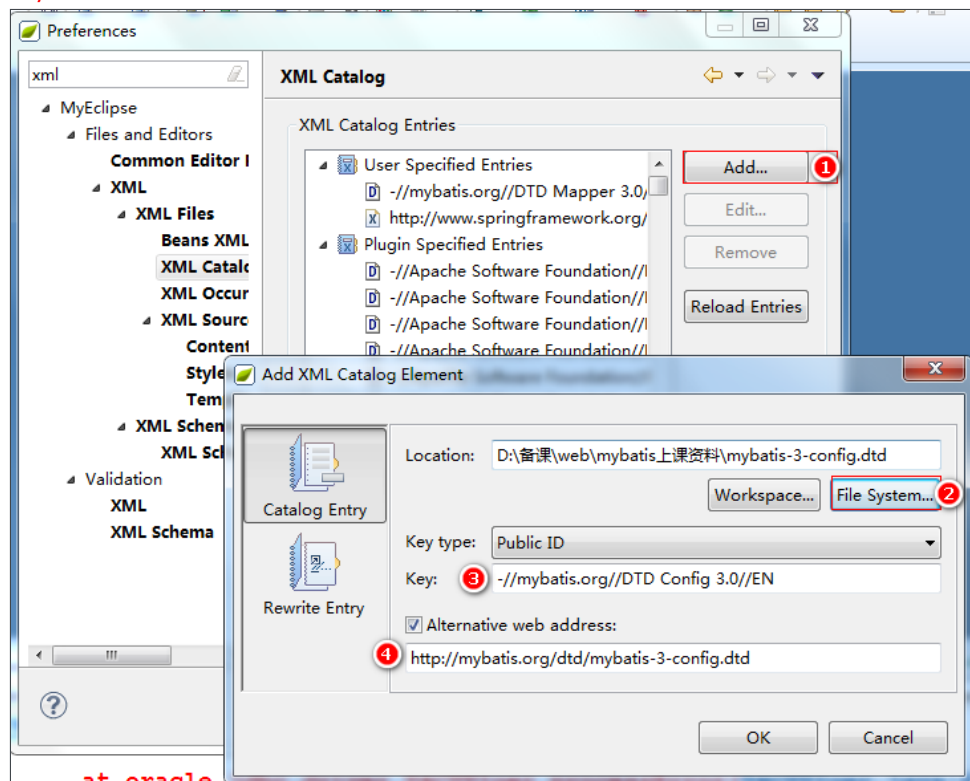
4.2.1. 点击Window---->找到Preferences



在搜索框中输入”xml”,并找到Xml Catalog 选项并点击右侧的Add..



4.2.2. 在弹出的选项卡中选择本地磁盘上mybatis配置文件的提示信息,并输入提示的key



注意:

主配置文件:

Public ID: `mybatis.org//DTD Config 3.0//EN`

Address: `http://mybatis.org/dtd/mybatis-3-config.dtd`

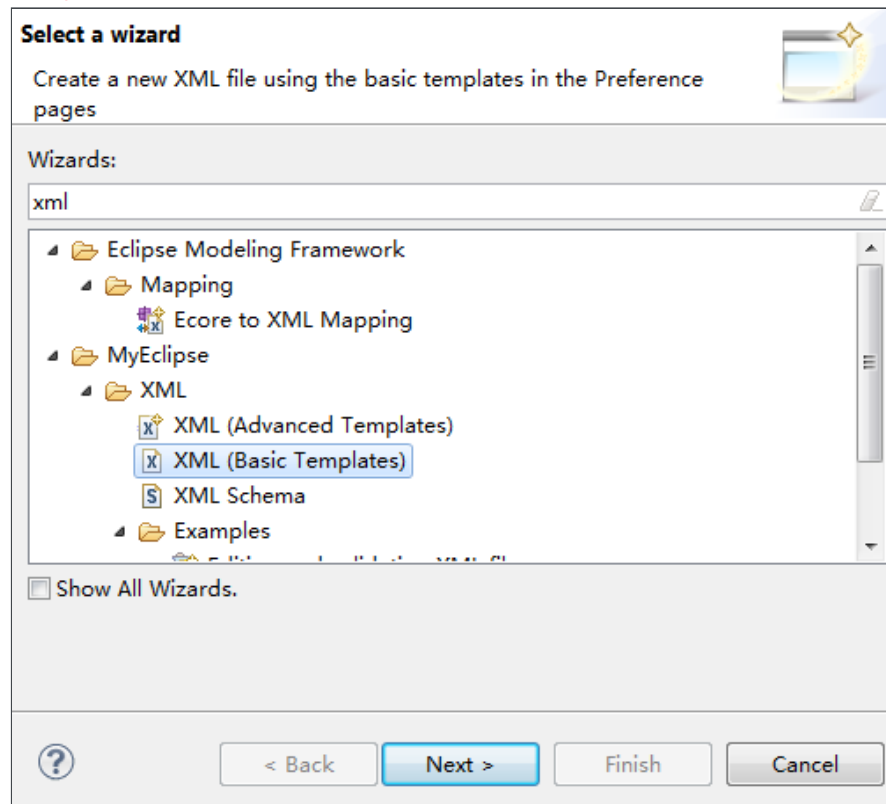
映射文件:

Public ID: `mybatis.org//DTD Mapper 3.0//EN`

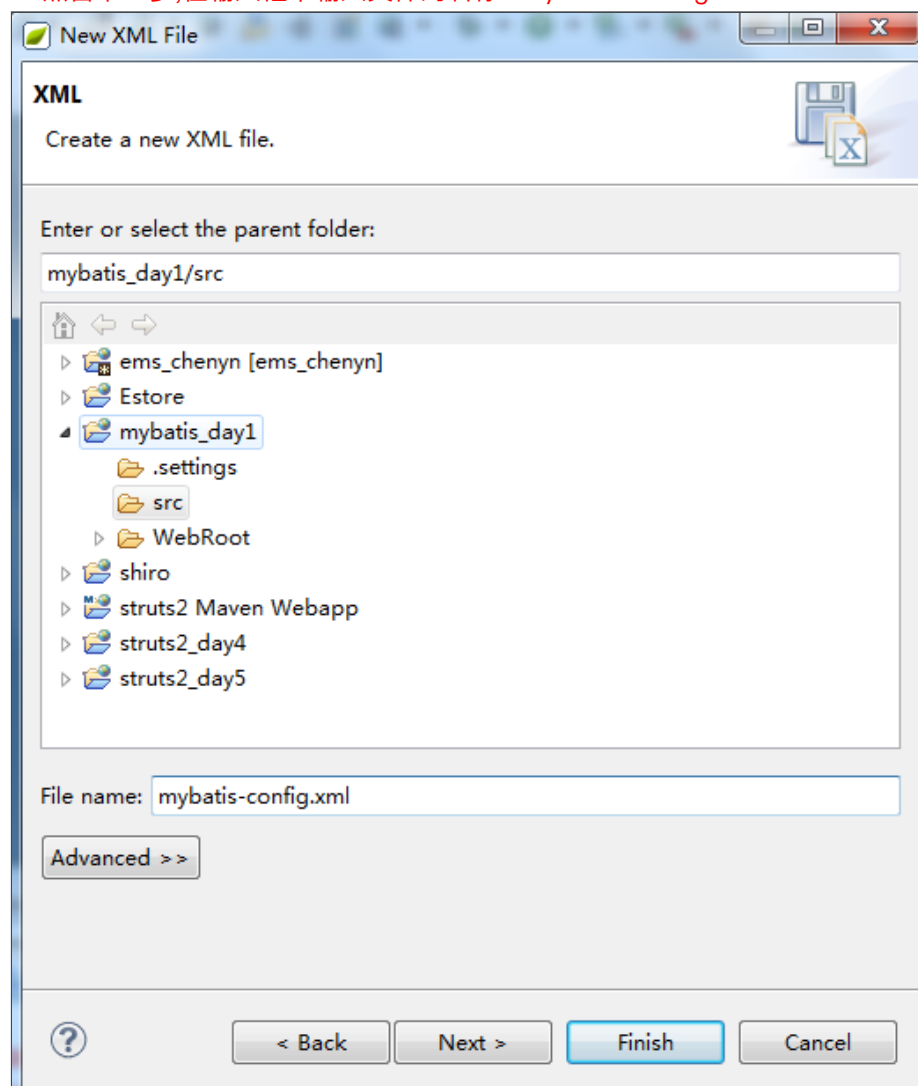
Address: `http://mybatis.org/dtd/mybatis-3-mapper.dtd`

4.3. 创建mybatis的主配置文件

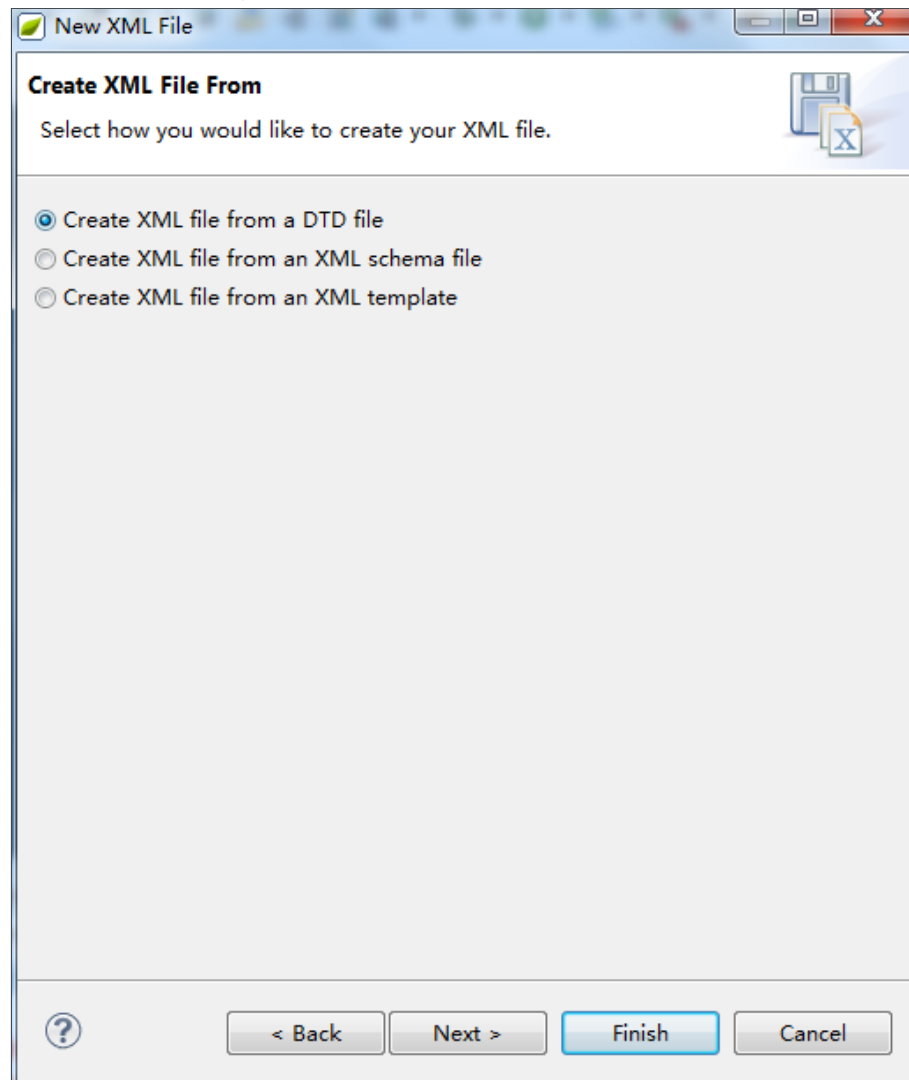
4.3.1. 在项目的src目录中鼠标右键,选择new,在搜索中搜索xml 选择XML Basic Template



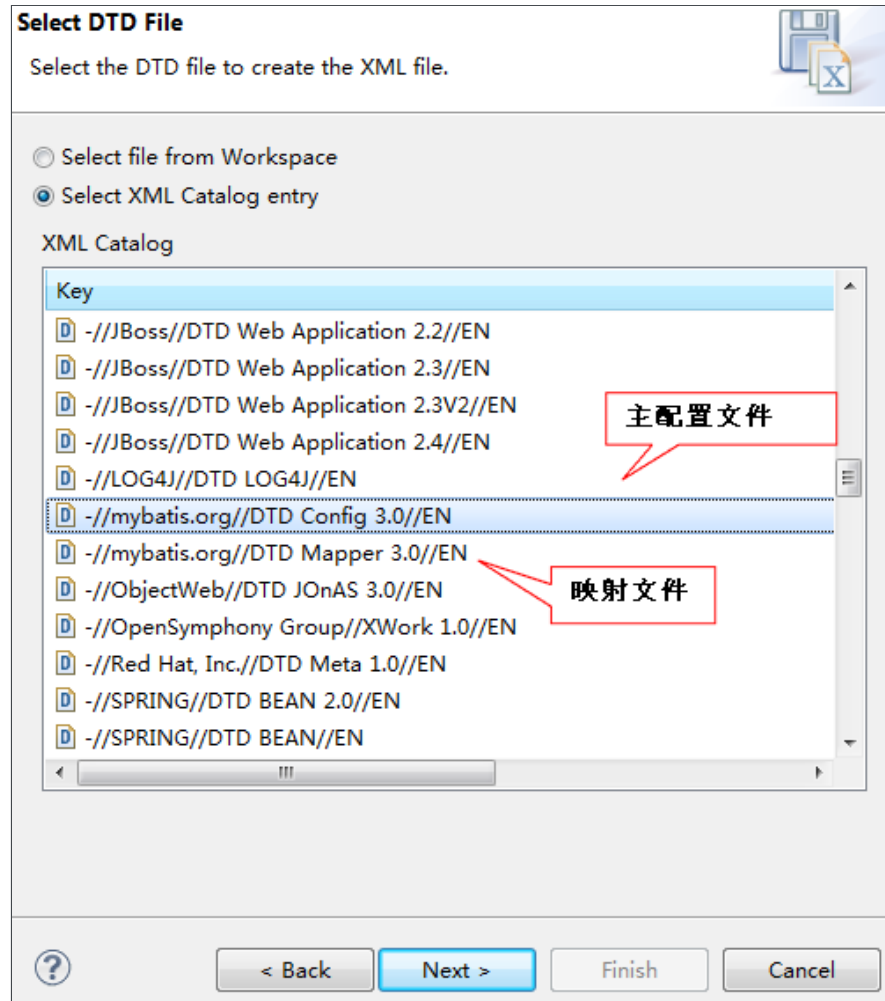
4.3.2. 点击下一步,在输入框中输入文件的名称 "mybatis-config.xml"



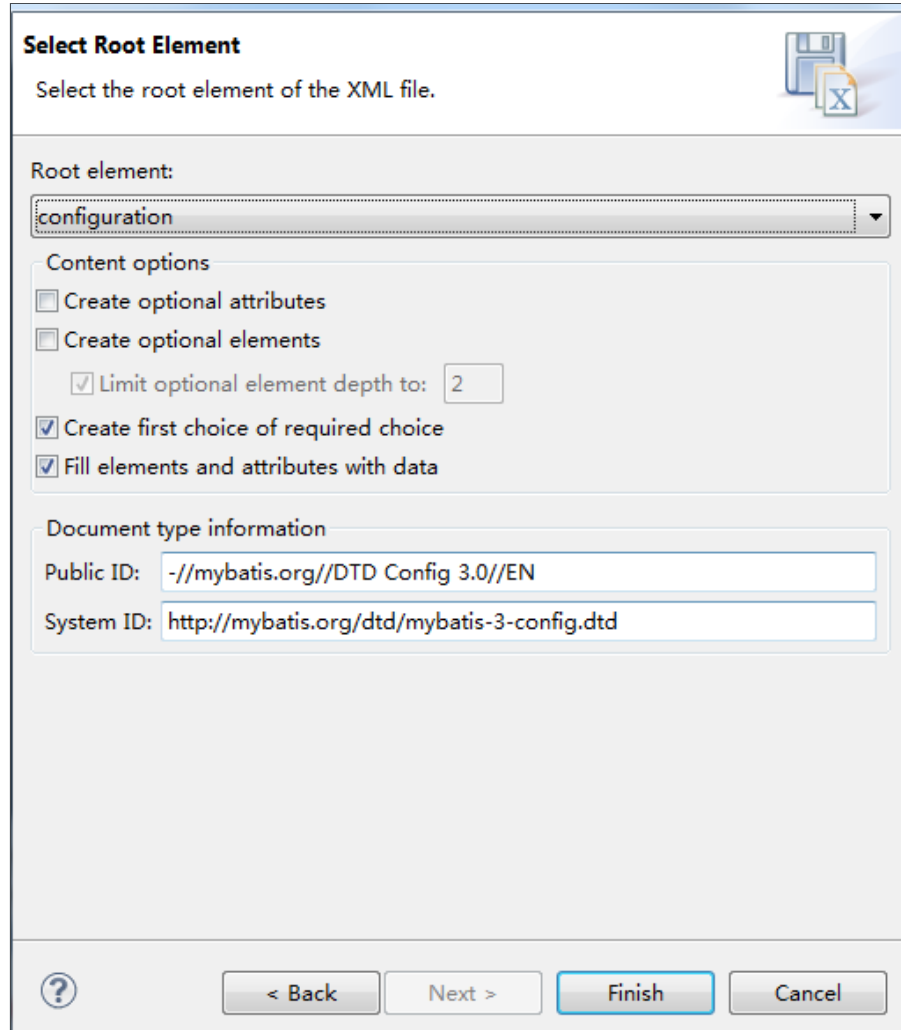
4.3.3. 点击Next 下一步,选择Create XML file from DTD file



4.3.4. 选择DTD文件,选择配置的主配置文件的dtd



4.3.5. 点击Next 下一步,直接Finish即可



The dialog box titled "Select Root Element" is used to configure the root element of an XML file. It includes a dropdown menu for the root element, a section for content options with checkboxes and a depth limit, and a section for document type information with text input fields. Navigation buttons at the bottom include Back, Next, Finish, and Cancel.

Select Root Element

Select the root element of the XML file.

Root element: configuration

Content options

- ☐ Create optional attributes
- ☐ Create optional elements
- ☒ Limit optional element depth to: 2
- ☒ Create first choice of required choice
- ☒ Fill elements and attributes with data

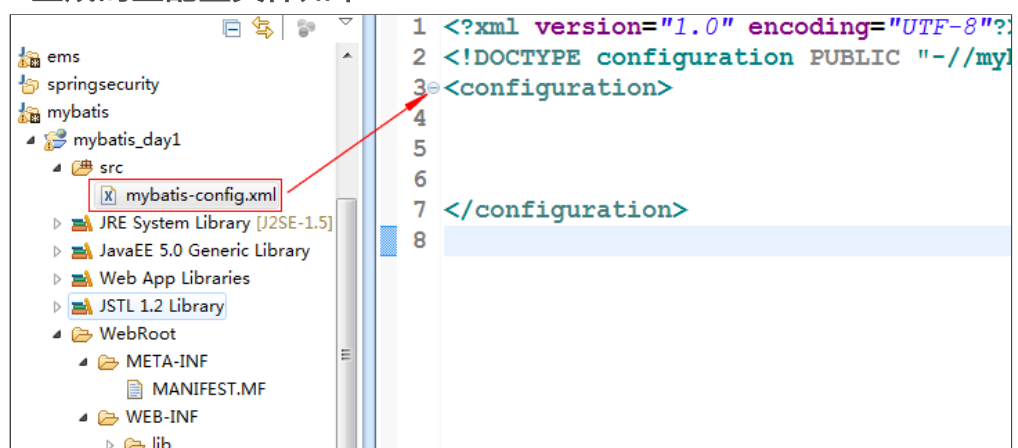
Document type information

Public ID: -//mybatis.org//DTD Config 3.0//EN

System ID: http://mybatis.org/dtd/mybatis-3-config.dtd

< Back Next > Finish Cancel

4.4. 生成的主配置文件如下



4.5. 书写连接相关的配置

```

<configuration>
  <!-- 环境标签
        default:默认使用的环境
    -->
  <environments default="dev">
    <!--
        id:环境的标识
    -->
    <environment id="dev">
      <!-- 使用jdbc的事务管理 -->
      <transactionManager type="jdbc"/>
      <dataSource type="POOLED">
        <property name="driver" value="oracle.jdbc.OracleDriver"/>
        <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe"/>
        <property name="username" value="hr"/>
        <property name="password" value="hr"/>
      </dataSource>
    </environment>
  </environments>
</configuration>

```

上边的key固定写死

4.6. 编写测试类获取连接

```

//读取文件
InputStream resourceAsStream = Resources.getResourceAsStream("mybatis-config.xml");
//创建sqlSessionFactory
SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder()
    .build(resourceAsStream);

//获取sqlSession
SqlSession openSession = sqlSessionFactory.openSession();
//调用一下方法测试是否正确
openSession.getConnection().commit();

```

4.7. 建表

```

create table t_user(
    id varchar2(40) primary key,
    name varchar2(40),
    age number(2)
);

insert into t_user values('1', '张三', 23);
insert into t_user values('2', '李四', 23);

```

4.8. 开发实体类

```

public class User {
    private String id;
    private String name;
    private Integer age;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

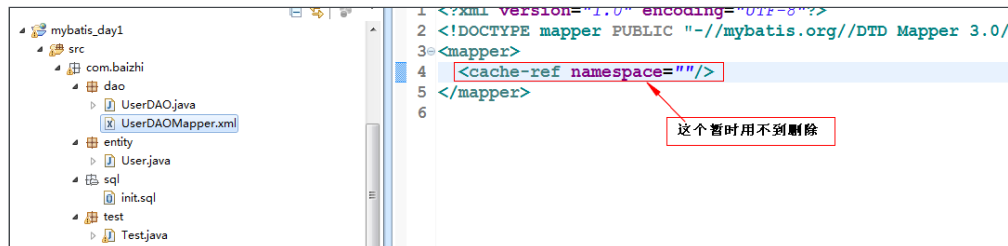
    public void setName(String name) {
        this.name = name;
    }
}

```


4.9. 开发DAO接口

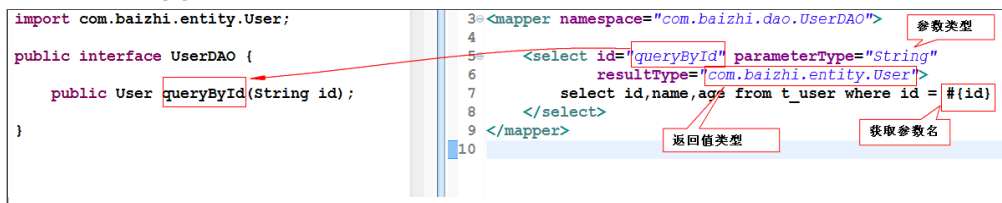
```
public interface UserDAO {  
  
    public User queryById(String id);  
  
}
```

4.10. 生成mapper映射文件

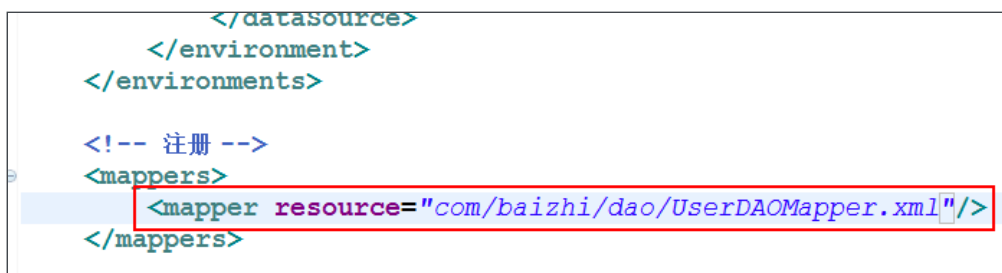


注意:创建mapper映射文件应该选择映射文件的提示

4.11. 编写mapper配置文件



4.12. 主配置文件中注册mapper配置文件



4.13. 测试DAO中的方法

```
//读取文件  
InputStream resourceAsStream = Resources.getResourceAsStream("mybatis-config.xml");  
//创建sqlSessionFactory  
SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder()  
    .build(resourceAsStream);  
  
//获取sqlSession  
SqlSession openSession = sqlSessionFactory.openSession();  
  
UserDAO mapper = openSession.getMapper(UserDAO.class);  
User queryById = mapper.queryById("1");  
System.out.println(queryById);
```

5. Mybatis中查询

5.1. 查询所有数据

```
public List<User> queryAll();  
  
<select id="queryAll" resultType="com.baizhi.entity.User">  
    select id,name,age from t_user  
</select>
```

如果查询结果为多个自动封装集合

注意:查询结果为多个时resultType同样书写包.类 自动封装集合

5.2. 多个参数的查询

```
public List<User> query(@Param("name") String name, @Param("age") Integer age);  
  
<select id="query" resultType="com.baizhi.entity.User">  
    select id,name,age from t_user where name = #{name} and age =#{age}  
</select>
```

@param中指定的值

注意:多个参数将省略parameterType,使用@param绑定参数

6. Mybaits中的增删改

6.1. 修改

```
public void update(User user);  
  
<update id="update" parameterType="com.baizhi.entity.User">  
    update t_user set name = #{name},age=#{age} where id = #{id}  
</update>
```

修改标签
方法名
实体类中的属性

6.2. 删除

```
public void delete(String id);  
  
<delete id="delete" parameterType="String">  
    delete from t_user where id = #{id}  
</delete>
```

删除标签

6.3. 添加

```
public void save(User user);  
<insert id="save" parameterType="com.baizhi.entity.User">  
    insert into t_user values (#{id},#{name},#{age})  
</insert>
```

类中的属性

7. Mybatis工具类封装

7.1. 封装工具类

```
package com.baizhi.util;  
  
import java.io.IOException;  
  
/**  
 * mybatis的工具类封装  
 * author:chenyn  
 * description:  
 * email:chenyn@zparkhr.com.cn  
 * datetime:2017年4月11日 下午3:25:06  
 */  
public class MybatisUtils {  
  
    //重量级资源 且线程安全  
    private static SqlSessionFactory sessionFactory;  
    //绑定ThreadLocal  
    private static final ThreadLocal<SqlSession> t = new ThreadLocal<SqlSession>();  
  
    static{  
        try {  
            InputStream resourceAsStream = Resources.getResourceAsStream("mybatis-config.xml");  
            sessionFactory= new SqlSessionFactoryBuilder().build(resourceAsStream);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
  
    /**  
     * 获取getSession  
     * description:  
     * @return  
     * datetime:2017年4月11日 下午3:26:50  
     */  
    public static SqlSession getSession(){  
        SqlSession sqlSession = t.get(); //获取线程连接  
        if(sqlSession==null){  
            sqlSession = sessionFactory.openSession();  
            t.set(sqlSession);  
        }  
        return sqlSession;  
    }  
  
    /**  
     * 释放资源  
     * description:  
     * datetime:2017年4月11日 下午3:26:34  
     */  
    public static void close(){  
        SqlSession sqlSession = t.get();  
        if(sqlSession!=null){  
            sqlSession.close();  
            t.remove();  
        }  
    }  
}
```