[今日课程大纲]

高级软件介绍(部分)

MySql 数据库建库建表语句强调

命名规范强调

基于 **MVC** 开发模式完成单表查询和新增

Eclipse 中项目默认发布路径

高级课程大纲介绍

框架是什么

MyBatis 简介

MyBatis 搭建流程

数据库连接池和 **JNDI** 复习

搭建流程详解(全局配置文件,**resultType** 原理及 **AutoMapping** 等)

MyBatis 三种查询方式

[知识点详解]

一.高级软件介绍

1. JDK 7
2. Eclipse mars2
3. MySql
4. Navicat

二.数据库 SQL 命令

1 创建数据库并指定编码

```
Create database  数据库名  default character set utf8
```

2.创建表

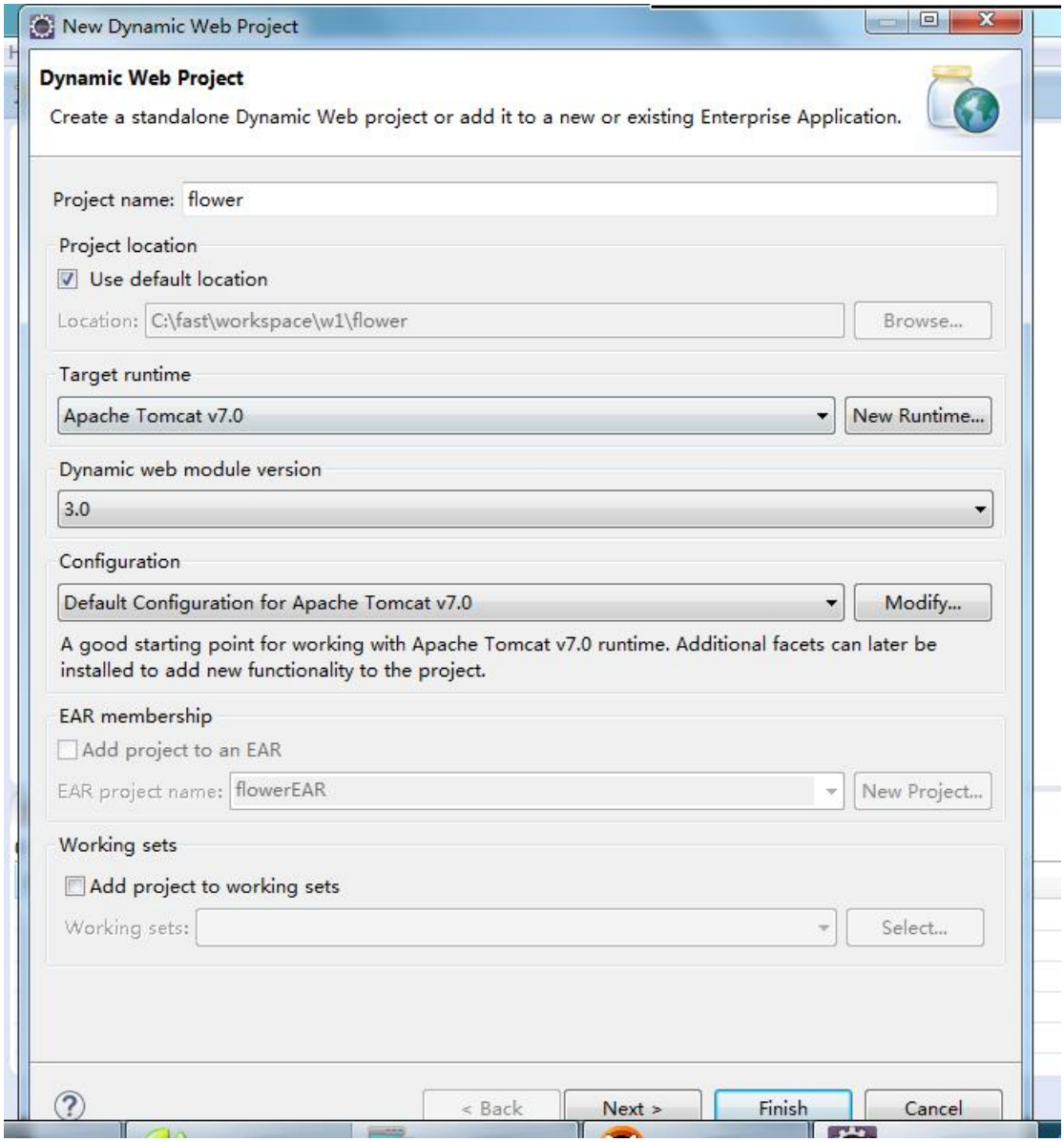
```
Create table  表名(
列名     类型    约束  auto_increment comment '备注',
);
```

三.Eclipse 使用

1. 创建项目

1.1 选择 target runtime,否则出现新建 jsp 报错

1.2 如果忘记选择,右键项目--> build path --> configure path -->选项卡 library --> 第四个 add library --> server runtime



2. Eclipse 默认会自己下载所需 tomcat 最简单结构.

三.命名规范

1.项目名:没有要求,不起中文

2.包:公司域名倒写 com.bjsxt

3.数据访问层:dao, persist, **mapper**

4.实体:entity, model, bean,javabean, **pojo**

5.业务逻辑: **service ,**biz

6.控制器: controller, **servlet**,action,web

7.过滤器: filter

8.异常: exception

9.监听器:listener

10.注释:

10.1 类上和方法上使用文档注释 /** */

10.2 在方法里面使用/* */ 或 //

11.类: 大驼峰

12.方法,属性:小驼峰

四.MVC 开发模式

1. M: Model 模型,实体类和业务和 dao
2. V: view 视图. JSP
3. C:Controller 控制器,servlet

3.1 作用:视图和逻辑分离

4. MVC 适用场景:大型项目开发.
5. 图示例

5.1 先设计数据库

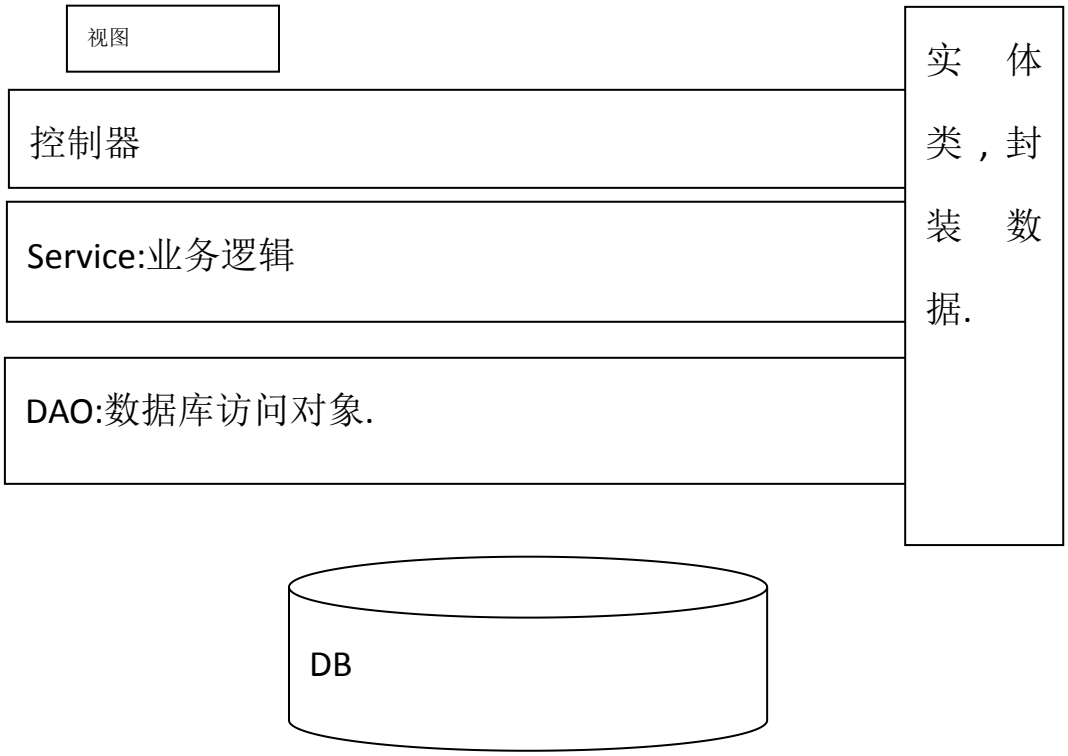
5.2 先写实体类

5.3 持久层

5.4 业务逻辑

5.5 控制器

5.6 视图



五.高级课程大纲介绍

1. 正课时间 46 天(9 周零 1 天)
2. 高级分为 3 部分:

2.1 第一部分:SSM 框架.11 天+5 天项目

2.1.1 MyBatis: 数据访问层框架

2.1.2 Spring 框架:IoC,AOP

2.1.3 SpringMVC 框架:对 Servlet 封装

2.2 第二部分:分布式项目开发(Ego) 6 天+14 天+5 天

2.3 第三部分:SSH 框架 5 天

六.框架是什么?

1. 框架:软件的半成品.未解决问题制定的一套约束,在提供功能基础上进行扩充.

2. 框架中一些不能被封装的代码(变量),需要使用框架者新建一个 xml 文件,在文件中添加变量内容.

2.1 需要建立特定位置和特定名称的配置文件.

2.2 需要使用 xml 解析技术和反射技术.

3. 常用概念

3.1 类库:提供的类没有封装一定逻辑.

举例:类库就是名言警句,写作文时引入名言警句

3.2 框架:区别与类库,里面有一些约束.

举例:框架是填空题

七.MyBatis 简介

1. Mybatis 开源免费框架.原名叫 iBatis,2010 在 google code,2013 年迁移到 github

2. 作用: 数据访问层框架.

2.1 底层是对 JDBC 的封装.

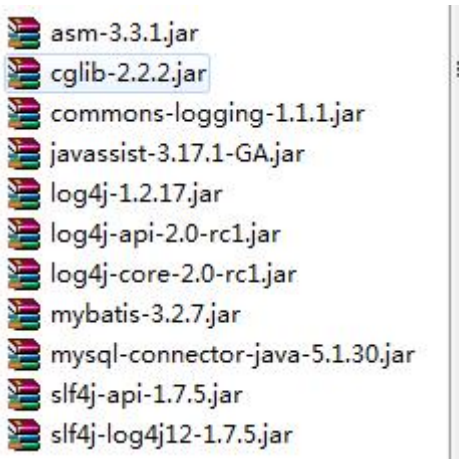
3. mybatis 优点之一:

3.1 使用 mybatis 时不需要编写实现类,只需要写需要执行的 sql 命

令

八. 环境搭建

1. 导入 jar



Cglib 依赖的包
动态代理包
日志包
字节码解析包也是 cglib 依赖的包
日志包
日志包
日志包
Mybatis 核心包
驱动
日志包

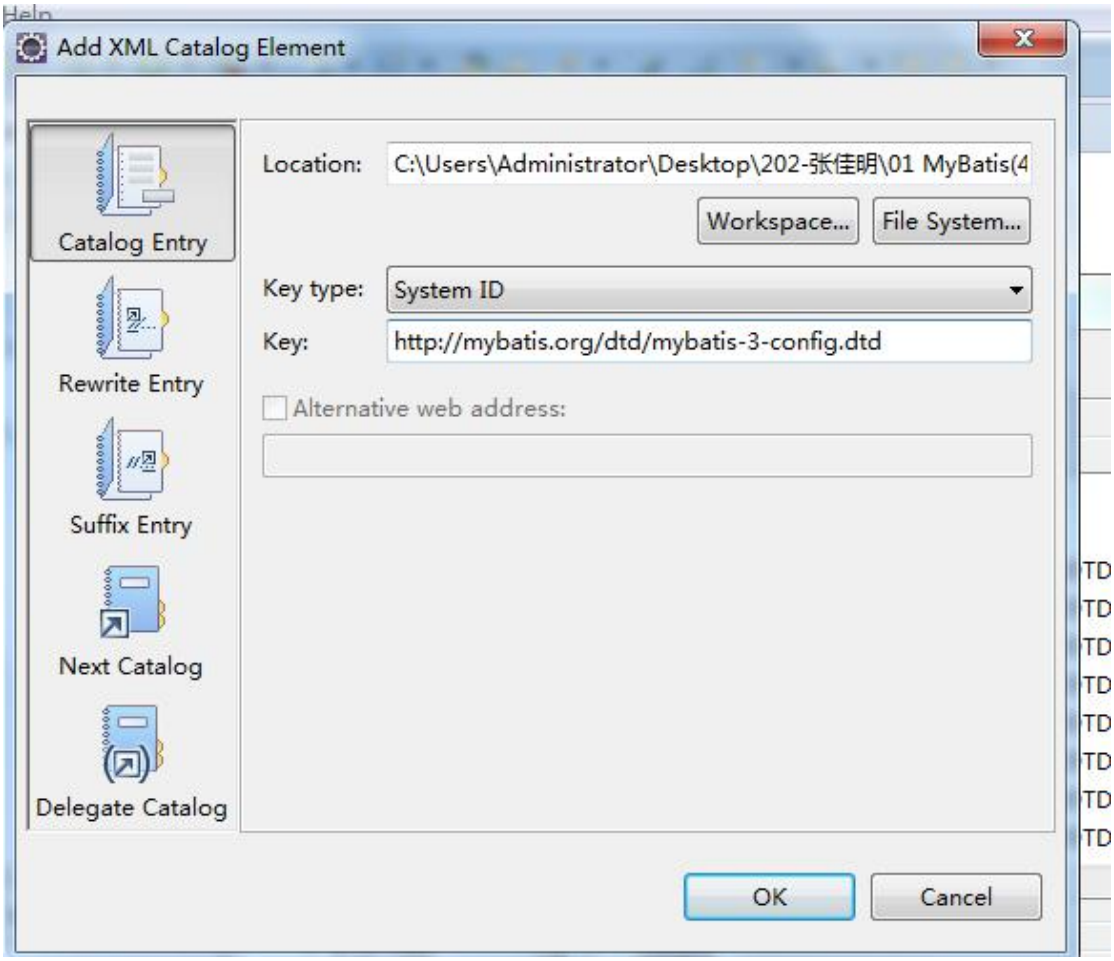
2. 在 src 下新建全局配置文件(编写 JDBC 四个变量)

2.1 没有名称和地址要求

2.2 在全局配置文件中引入 DTD 或 schema

2.2.1 如果导入 dtd 后没有提示

Window--> preference --> XML --> XMl catalog --> add 按钮



2.3 全局配置文件内容

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!-- default 引用 environment 的 id,当前所使用的环境 -->
    <environments default="default">
        <!-- 声明可以使用的环境 -->
        <environment id="default">
            <!-- 使用原生 JDBC 事务 -->
            <transactionManager type="JDBC"></transactionManager>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://localhost:3306/ssm"/>
                <property name="username" value="root"/>
                <property name="password" value="smallming"/>
            </dataSource>
        </environment>
    </environments>
```

```
    <mappers>
        <mapper resource="com/bjsxt/mapper/FlowerMapper.xml"/>
    </mappers>
</configuration>
```

3. 新建以 mapper 结尾的包,在包下新建:实体类名+Mapper.xml

3.1 文件作用:编写需要执行的 SQL 命令

3.2 把 xml 文件理解成实现类.

3.3 xml 文件内容

```
<?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">
<!-- namesapce:理解成实现类的全路径(包名+类名) -->
<mapper namespace="a.b" >
   <!-- id:方法名
      parameterType:定义参数类型
      resultType:返回值类型.

      如果方法返回值是 list,在 resultType 中写 List 的泛型,
因为 mybatis
      对 jdbc 封装,一行一行读取数据
   -->
   <select id="selAll"
```

```
resultType="com.bjsxt.pojo.Flower">
      select * from flower
   </select>
</mapper>
```

4. 测试结果(只有在单独使用 mybatis 时使用,最后 ssm 整合时下面代码不需要编写.)

```
InputStream is =
Resources.getResourceAsStream("myabtis.xml");
      //使用工厂设计模式
      SqlSessionFactory factory = new
SqlSessionFactoryBuilder().build(is);
      //生产 SqlSession
      SqlSession session=factory.openSession();

      List<Flower> list =
session.selectList("a.b.selAll");
      for (Flower flower : list) {
         System.out.println(flower.toString());
      }

      session.close();
```

九. 环境搭建详解

1.全局配置文件中内容

1.1 <transactionManager/> type 属性可取值

1.1.1 JDBC,事务管理使用 JDBC 原生事务管理方式

1.1.2 MANAGED 把事务管理转交给其他容器.原生 JDBC 事务 setAutoMapping(false);

1.2 <dataSouce/>type 属性

1.2.1 POOLED 使用数据库连接池

1.2.2 UNPOOLED 不实用数据库连接池,和直接使用 JDBC 一样

1.2.3 JNDI :java 命名目录接口技术.

十.数据库连接池

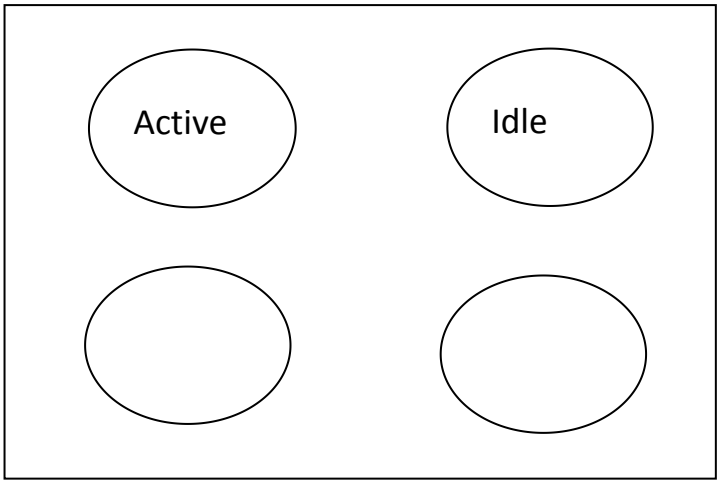
1.在内存中开辟一块空间,存放多个数据库连接对象.

2.JDBC Tomcat Pool,直接由 tomcat 产生数据库连接池.

3.图示

3.1 active 状态:当前连接对象被应用程序使用中

3.2 Idle 空闲状态:等待应用程序使用



4.使用数据库连接池的目的

4.1 在高频率访问数据库时,使用数据库连接池可以降低服务器系统压力,提升程序运行效率.

4.1.1 小型项目不适用数据库连接池.

5.实现 JDBC tomcat Pool 的步骤.

5.1 在 web 项目的 META-INF 中存放 context.xml,在 context.xml 编写数据库连接池相关属性

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
   <Resource
      driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://localhost:3306/ssm"
      username="root"
```

```
      password="smallming"
      maxActive="50"
      maxIdle="20"
      name="test"
      auth="Container"
      maxWait="10000"
      type="javax.sql.DataSource"
   />
</Context>
```

5.2 把项目发布到 tomcat 中,数据库连接池产生了

6.可以在 java 中使用 jndi 获取数据库连接池中对象

6.1 Context:上下文接口.context.xml 文件对象类型

6.2 代码:

```
Context cxt = new InitialContext();
        DataSource ds = (DataSource)
cxt.lookup("java:comp/env/test");
        Connection conn = ds.getConnection();
```

6.3 当关闭连接对象时,把连接对象归还给数据库连接池,把状态改变成 Idle

十一. 三种查询方式

1.selectList() 返回值为 List<resultType 属性控制>

1.1 适用于查询结果都需要遍历的需求

```
List<Flower> list = session.selectList("a.b.selAll");
      for (Flower flower : list) {
         System.out.println(flower.toString());
      }
```

2.selectOne() 返回值 Object,

2.1 适用于返回结果只是变量或一行数据时

```
      int count = session.selectOne("a.b.selById");
      System.out.println(count);
```

3.selectMap() 返回值 Map

3.1 适用于需要在查询结果中通过某列的值取到这行数据的需求.

3.2 Map<key,resultType 控制>

```
Map<Object, Object> map = session.selectMap("a.b.c",
"name123");
      System.out.println(map);
```