# 第四周总结：

## 大致总结

### (一)xx内部培训

在周一周二两天的时间里，接受了短期的xxx内部培训。讲师的专业让整个课程声动有趣并且学到了很多东西，团队间紧密的协作一次次凝聚了每个人。时间虽然短，却让自己对公司的发展、文化、未来等有了全面的认识，对自己在未来的工作有了启发，并且作为一个过渡，开始融入这个团体。

xxx在发展道路上的艰辛让人为之动容，逐步形成的企业文化和工作法让人看到公司多年的积淀影响着一代一代xx人，对未来的憧憬让人看到神州不断创新与远见。

在培训中让人印象深刻的是xxx工作法中进取篇中的有关职场学习的特点与方法的介绍。它认为：是以解决问题为导向的成年人学习。对于初入职场的小白来说，如何高效进取十分重要，学校的以课本为核心的学习法早已不再适用。在工作中，最终的目的在于解决问题，为xxx贡献最大的价值才能体现自己价值。因此，在今后，需以如何解决问题为核心开展工作和学习，在不断实践中吸取经验，不断为xxx创造最大化的价值，同时提升个人的能力，为自己的职业生涯不断添砖加瓦。

另外，在沟通上是自己需要不断学习总结的。不论在与上级还是在团队成员之间的沟通都需要高效及时，以合适的方式去表达自己的想法，主动去承担责任，做一个有担当让人信任的伙伴。

### (二) 学习Springmvc与Mybatis

对这两个框架进行学习后，基本掌握其使用方法，了解了设计思想和架构，对其做了相关的笔记，对框架的使用更加通透了。希望在今后实战中不断强化对框架的理解，积累更多的开发经验，如果时机成熟，希望能深入理解其设计思想和底层原理。

## 学习笔记

包含Springmvc、Mybatis学习笔记

### (一)Springmvc

\*\*\*\*\*\*springmvc简单介绍\*\*\*\*\*\*

#### 一、入门程序

1.动态工程

2.导包

3.配置文件

处理器(springmvc.xml Controller扫描包+注解驱动)+前端控制器(web.xml DispaccherServlet)

4.创建jsp 创建pojo 创建controller

#### 二、架构分析

一个中心

前端控制器

三个基本点(三大组件)

处理器映射器、处理器适配器、视图解析器

需要用户开发的组件有handler、view

#### 三、参数绑定

1.默认支持参数类型

HttpServletRequest HttpServletResponse HttpSession

2.简单类型

基本数据类型(推荐使用包装类型)

整形：Integer、int

字符串：String

单精度：Float、float

双精度：Double、double

布尔型：Boolean、boolean

3.pojo类型

其实就是与页面表单对应的bean

Items items input name=name Items对象中的属性名一致

要求：pojo对象中的属性名和表单中input的name属性一致。

4.包装pojo类型

QueryVo(里面Items) QueryVo vo input name=items.name

要求：QueryVo对象中的属性名和表单中input的name属性一致。

5.自定义参数绑定

日期的转换

一般使用<mvc:annotation-driven/>注解驱动加载处理器适配器

#### 四、springmvc整合mybatis

需要的jar包

1.spring（包括springmvc）

2.mybatis

3.mybatis-spring整合包

4.数据库驱动

5.第三方连接池。

整合思路

Dao层：

1、SqlMapConfig.xml，空文件即可，但是需要文件头。

2、applicationContext.xml

a) 数据库连接池

b) SqlSessionFactory对象，需要spring和mybatis整合包下的。

c) 配置mapper文件扫描器。

Service层：

1、applicationContext-service.xml包扫描器，扫描@service注解的类。

2、applicationContext-trans.xml配置事务。

Controller层：

1、Springmvc.xml

a) 包扫描器，扫描@Controller注解的类。

b) 配置注解驱动

c) 配置视图解析器

Web.xml文件：

1、配置spring

2、配置前端控制器。

#### 五、Springmvc与Struts2区别

1.入口 servlet Filter

2.实例化 单例 多例

3.返回值 Request域 值栈

\*\*\*\*\*\*springmvc深入介绍\*\*\*\*\*\*

#### 六、高级数据绑定

数组

Integer[] ids name="ids"(多个) value="${item.id }"

List

由于不能直接绑定List类型，需进行包装

QueryVo vo QueryVo{List<item> itemList}

name="itemList[index]}.name"

#### 七、Controller层方法返回值

ModelAndView 无敌的 带数据 返回视图路径

String 返回视图路径 modle带数据(推荐 解耦)

void

#### 八、springmvc上传图片步骤总结

1.加入包

文件上传包

io流包

2.虚拟目录

D:\upload /pic

3.表单书写

1)enctype="multipart/form-data"

2)<img src="/pic/${item.pic}" width=100 height=100/>

此处前缀'src'必须同虚拟目录一致，不然访问不到图片

3)<input type="file" name="pictureFile"/>

4.在springmvc.xml中配置文件上传解析器

<!-- 文件上传,id必须设置为multipartResolver -->

<bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

<!-- 设置文件上传大小 -->

<property name="maxUploadSize" value="5000000" />

</bean>

5.在controller中获取并上传和更新数据库

1)MultipartFile pictureFile

名称必须与input中的相同

2)设置图片名称

String picName = UUID.randomUUID().toString();

3)获取文件后缀名

String extName=FilenameUtils.getExtension(pictureFile.getOriginalFilename());

4) 开始上传

pictureFile.transferTo(new File("D:\\upload\\" + picName +"."+ extName));

此处文件写法还可为："D:/upload/"

5)更新数据库

// 设置图片名到商品中

item.setPic(picName + extName);

// 更新商品

this.itemService.updateItemById(item);

6)处理页面

#### 九、json数据交互(springmvc框架下)

环境要求：如果需要springMVC支持json，必须加入json的处理jar

页面：

$(function(){

//此处注意需用''来包裹，才能称之为json串

var params = '{"id": 1,"name": "测试商品","price": 99.9,"detail": "测试商品描述","pic": "123456.jpg"}';

$.ajax({

url : "${pageContext.request.contextPath }/json.action",

data : params,//发生数据

contentType : "application/json;charset=UTF-8",//发送数据的格式

type : "post",

dataType : "json",//回调

success : function(data){

alert(data.name);

}

});

});

控制器：

此处注意对@RequestBody和@ResponseBody的使用

public @ResponseBody Item testJson(@RequestBody Item item) {

return item;//由于ResponseBody的注解，直接将Item转换为json格式返回前端

}

@RequestBody注解实现接收http请求的json数据，将json数据转换为java对象进行绑定

@ResponseBody注解实现将Controller方法返回java对象转换为json响应给客户端

#### 十、拦截器——验证是否登录

处理流程

1、有一个登录页面，需要写一个Controller访问登录页面

2、登录页面有一提交表单的动作。需要在Controller中处理。

a) 判断用户名密码是否正确（在控制台打印）

b) 如果正确,向session中写入用户信息（写入用户名username）

c) 跳转到商品列表

3、拦截器。

a) 拦截用户请求，判断用户是否登录（登录请求不能拦截）

b) 如果用户已经登录。放行

c) 如果用户未登录，跳转到登录页面。

需要注意的是：

a)登录请求login.action不能拦截

String requestURI = request.getRequestURI();

if(!requestURI.contains("/login"))

b)放行问题：

根据session中的user是否为空来判断

### （二）Mybatis

\*\*\*\*\*\*mybatis简单介绍\*\*\*\*\*\*

#### 一、MyBatis简介

MyBatis是一个优秀的持久层框架，它对jdbc的操作数据库的过程进行封装，

使开发者只需要关注 SQL 本身，而不需要花费精力去处理例如注册驱动、

创建connection、创建statement、手动设置参数、结果集检索等jdbc繁杂

的过程代码。

Mybatis通过xml或注解的方式来处理数据库

#### 二、环境搭建

1.jar包

mybatis核心包、依赖包、数据驱动包

2.配置文件

log4j.properties

SqlMapConfig.xml(引入N个user.xml文件)

<mapper resource="user">

3.书写user.xml

<mapper namespace="user">

4.创建pojo——User.java，与user.xml对应

5.测试

a）加载SqlMapConfig.xml

InputStream inputStream = getResourceAsStream("SqlMapConfig.xml");

b)创建SqlSessionFactory对象并读取SqlMapConfig.xml

SqlSessionFactoryBuilder sqlSessionFactoryBuilder = new SqlSessionFactoryBuilder();

SqlSessionFactory sqlSessionFactory = sqlSessionFactoryBuilder.build(inputStream);

c)获取SqlSession对象

SqlSession sqlsession = sqlSessionFactory.openSession();

d)执行sql语句

User user = sqlSession.selectOne("user.findUserById", 10);

e)事物管理

注意：增删改都需要此步骤，查询不需要

sqlSession.commit();

sqlSession.close();

#### 三、parameterType和resultType

parameterType：指定输入参数类型，mybatis通过ognl从输入对象中获取参数值拼接在sql中。

resultType：指定输出结果类型，mybatis将sql查询结果的一行记录数据映射为resultType指定类型的对象。如果有多条数据，则分别进行映射，并把对象放到容器List

#### 四、#{}与${}区别

#{} 占位符

username like '%'#{}'%'

${} 字符串拼接

username like '%#{}%'

#### 五、 Mybatis解决jdbc编程的问题

1、 数据库连接创建、释放频繁造成系统资源浪费从而影响系统性能，如果使用数据库连接池可解决此问题。

解决：在SqlMapConfig.xml中配置数据连接池，使用连接池管理数据库链接。

2、 Sql语句写在代码中造成代码不易维护，实际应用sql变化的可能较大，sql变动需要改变java代码。

解决：将Sql语句配置在XXXXmapper.xml文件中与java代码分离。

3、 向sql语句传参数麻烦，因为sql语句的where条件不一定，可能多也可能少，占位符需要和参数一一对应。

解决：Mybatis自动将java对象映射至sql语句，通过statement中的parameterType定义输入参数的类型。

4、 对结果集解析麻烦，sql变化导致解析代码变化，且解析前需要遍历，如果能将数据库记录封装成pojo对象解析比较方便。

解决：Mybatis自动将sql执行结果映射至java对象，通过statement中的resultType定义输出结果的类型。

#### 六、Mapper动态代理开发

遵循四个原则

接口 方法名 == User.xml 中 id 名

返回值类型 与 Mapper.xml文件中返回值类型要一致

方法的入参类型 与Mapper.xml中入参的类型要一致

命名空间 绑定此接口

示例：

Mapper接口

public User findUserById(Integer id);

User.xml：

<mapper namespace="com.itheima.mybatis.mapper.UserMapper">//命名空间绑定

<select id="findUserById" parameterType="Integer" resultType="com.itheima.mybatis.pojo.User">

select \*from

user where id=#{value }

</select>

</mapper>

\*\*\*\*\*\*mybatis深入介绍\*\*\*\*\*\*

#### 七、使用resultMap进行手动映射

使用场合：

当表字段与pojo属性字段不一致时

如果一致，不用写

示例：

<!-- resultMap最终还是要将结果映射到pojo上，type就是指定映射到哪一个pojo -->

<!-- id：设置ResultMap的id -->

<resultMap type="order" id="orderResultMap">

<!-- 定义主键 ,非常重要。如果是多个字段,则定义多个id -->

<!-- property：主键在pojo中的属性名 -->

<!-- column：主键在数据库中的列名 -->

<id property="id" column="id" />

<!-- 定义普通属性 -->

<result property="userId" column="user\_id" />

</resultMap>

<!-- 查询所有的订单数据 -->

<select id="queryOrderAll" resultMap="orderResultMap">

SELECT id, user\_id,

number,

createtime, note FROM `order`

</select>

#### 八、动态sql

1、if where标签

示例说明：根据用户名和性别进行查询

示例书写：

select

\*from user

<!--where标签可以自动添加where，同时处理sql语句中第一个and关键字-->

<where>

<if test="sex!=null and sex!=''">

sex = #{sex}

</if>

<if test="username!=null and username!=''">

<!--模糊查询-->

and username like '%${username}%'

</if>

</where>

示例要点：使用where去除第一个and关键字 注意and的位置(字段前)

2、foreach标签

向sql传递数组或List，mybatis使用foreach解析

示例说明：

根据多个id查询用户信息

查询sql：SELECT \* FROM user WHERE id IN (1,10,24)

示例书写：

select

\*from user

where id in

<foreach collection="list" item="id" separator="," open="(" close=")">

#{id}

</foreach>

示例要点：a)在传递collection值时，如果List或Array没有被封装成QueryVo的属性,只能写list或array

b)可以把id in放在open属性中 open="id in("

#### 九、关联查询

核心理解：

a)谁为中心谁就是一

即关联对象加在它的属性上，映射就写在它的映射文件里

b）书写resultMap映射时，需要的字段都必须要求映射

1、一对一关联

使用resluttype(不推荐)

使用resultMap

association ：嵌套配置一对一属性

示例说明：

需求：查询所有订单信息，关联查询下单用户信息

示例书写：

a）改造Orders 加入private User user

b）手动映射resultMap 需要什么字段就映射什么字段

<!-- type ：返回值的全限定类名，或类型别名 -->

<resultMap type="order" id="orderUserResultMap">

<id property="id" column="id" />

<result property="userId" column="user\_id" />

<result property="number" column="number" />

<result property="createtime" column="createtime" />

<result property="note" column="note" />

<!-- association ：配置一对一属性 -->

<!-- property:order里面的User属性名 -->

<!-- javaType:属性类型 -->

<association property="user" javaType="user">

<!-- id:声明主键，表示user\_id是关联查询对象的唯一标识-->

<id property="id" column="user\_id" />

<result property="username" column="username" />

<result property="address" column="address" />

</association>

</resultMap>

c)书写查询语句

<!-- 一对一关联，查询订单，订单内部包含用户属性 -->

<select id="queryOrderUserResultMap" resultMap="orderUserResultMap">

SELECT

o.id,

o.user\_id,

o.number,

o.createtime,

o.note,

u.username,

u.address

FROM

`order` o

LEFT JOIN `user` u ON o.user\_id = u.id

</select>

示例要点：

a)改造Orders 加入User user 时，注意加入getter和setter

b)书写resultMap映射时，需要的字段都必须要求映射

2、一对多关联

使用resluttype(不推荐)

使用resultMap

collection ：嵌套配置一对多属性

示例说明：

需求：查询所有用户信息及用户关联的订单信息

示例书写：

a）改造User 加入List<Order> orders

b）手动映射resultMap 需要什么字段就映射什么字段

<!-- type ：返回值的全限定类名，或类型别名 -->

<resultMap type="user" id="userOrderResultMap">

<id property="id" column="id" />

<result property="username" column="username" />

<result property="birthday" column="birthday" />

<result property="sex" column="sex" />

<result property="address" column="address" />

<!-- 配置一对多的关系 -->

<collection property="orders" javaType="list" ofType="order">

<!-- 配置主键，是关联Order的唯一标识 -->

<id property="id" column="oid" />

<result property="number" column="number" />

<result property="createtime" column="createtime" />

<result property="note" column="note" />

</collection>

</resultMap>

c)书写查询语句

<!-- 一对多关联，查询订单同时查询该用户下的订单 -->

<select id="queryUserOrder" resultMap="userOrderResultMap">

SELECT

u.id,

u.username,

u.birthday,

u.sex,

u.address,

o.id oid,

o.number,

o.createtime,

o.note

FROM

`user` u

LEFT JOIN `order` o ON u.id = o.user\_id

</select>

示例要点：

a)改造User 加入List<Order> orders，注意加入getter和setter,利用List存多个订单

b)注意collection标签中其对象类型：javaType="list" ofType="order"

在association为：javaType="user"

#### 十、Mybatis整合spring

整合思路

1、SqlSessionFactory对象应该放到spring容器中作为单例存在。

2、传统dao的开发方式中，应该从spring容器中获得sqlsession对象。

3、Mapper代理形式中，应该从spring容器中直接获得mapper的代理对象。

4、数据库的连接以及数据库连接池事务管理都交给spring容器来完成。