**今天完成的事情：**

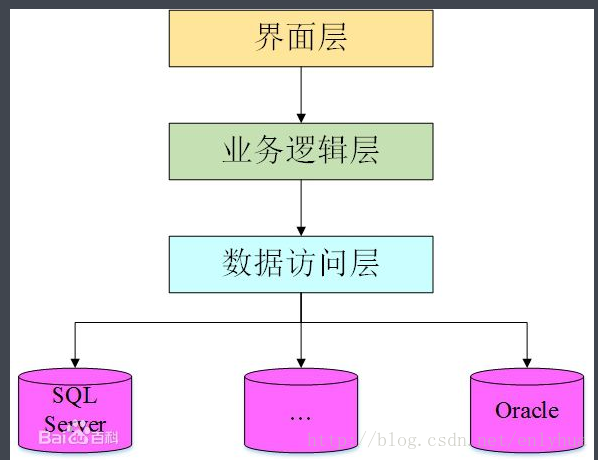
1. **任务总结**

**任务名称**：java-task1

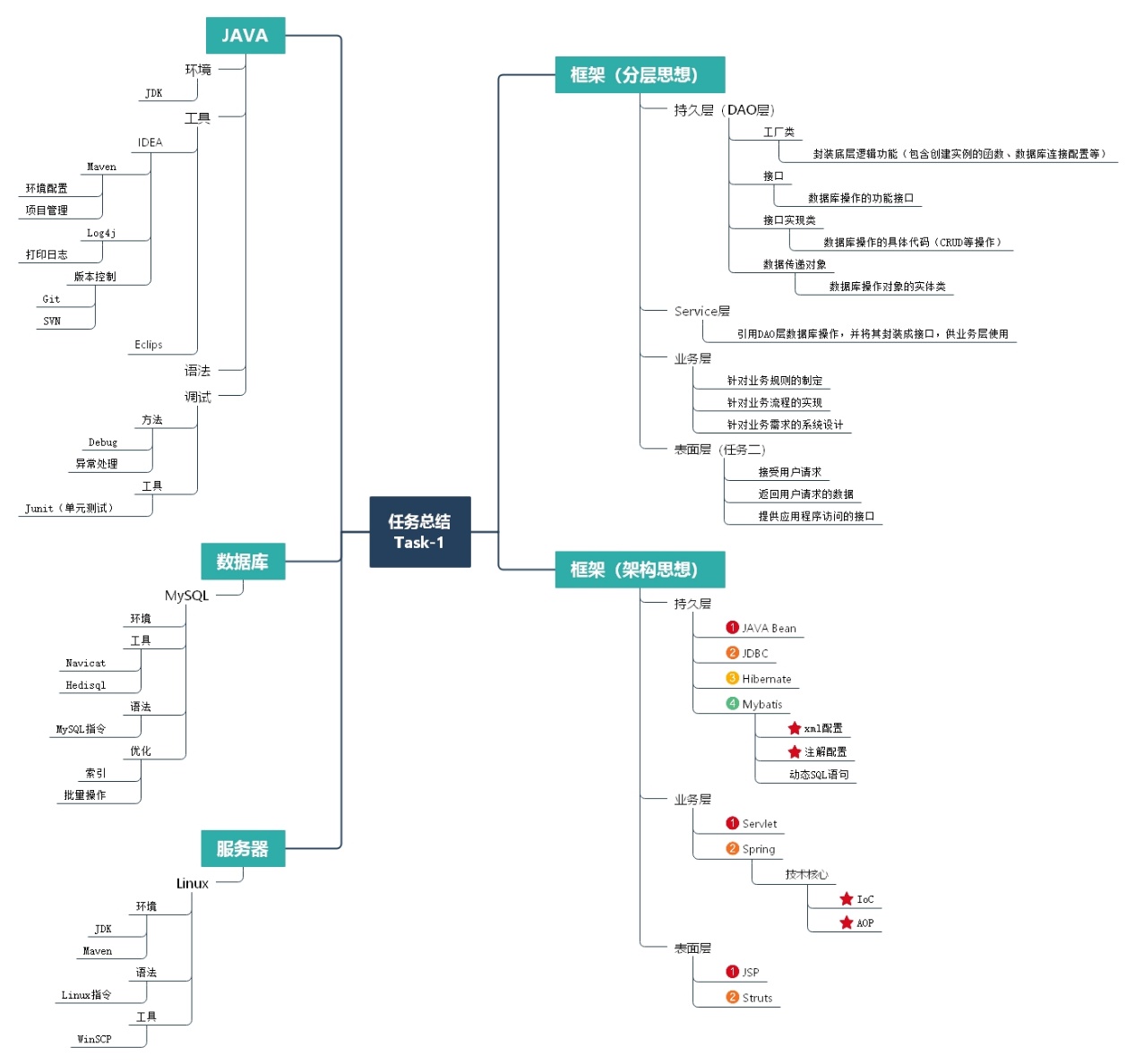
**成果连接**：<https://github.com/JrongXu/Task-1>

**任务耗时**：2019.3.15-2019.4.15（零基础）

对于JAVA程序框架，基本可以分成如图所示的结构



以下为任务总结，技能脑图：



1. 首先进行任务1.1-1.16的学习，配置好数据库，使用Navicat，学习MySQL语句进行增删改查的操作。然后下载JDK配置JAVA环境，下载IDEA配置Maven环境，使用Maven的clean，install操作。这之前的操作是简单地
2. 从这里进行1.17-1.21的学习，这里开始真正学习JAVA编程，学习使用框架进行对接数据库的增删改查操作。我在这里花费了大量的时间，当然因为我的效率低也是原因之一。
3. 开始搭建底层构架之前，先开始边补基础知识边学习JDBC构件知识，由于JDBC也有一些非底层的封装方法（例如加载驱动，加载数据库对象，创建对象等），所以在一开始构建JDBC时很难理解这些概念，就又回过头去补基础知识。
4. 在模仿搭建JDBC连接数据库完成CRUD操作之后，开始接触JDBCtemplate和Mybatis的概念，同时也开始接触Spring框架。在理解Spring核心概念IoC和AOP的过程中被弄迷糊，学习了很久（现在想想两大核心并没有用到，应该先构建框架，逐渐在代码运作中理解）。
5. 尝试构建Spring的JDBCtemplate的方式连接数据库，Spring的JDBCtemplate对JDBC的数据库操作进行了封装，并支持AOP的声明事务，内嵌于Spring框架，在Spring的xml文件中配置对数据库的操作
6. 接着学习Mybatis的搭建，Mybatis是持久层的轻量级框架，可以以xml和注解+反射的技术，对DAO层进行配置和操作。将对数据库的操作封装在映射文件mapper中，使用工厂模式的概念来创建数据库操作对象，执行对数据库的操作。
7. JDBC和Spring的JDBCtemplate和Mybatis的区别
8. **JDBC**

我认为JDBC是简单地通过Java API对数据库访问，包含例如事务控制、资源管理、异常处理等事务管理

由一系列接口方法完成对数据库的操作

1# 通过接口方法，加载JDBC数据库驱动

2# 通过接口方法，建立数据库连接Connection

3# 通过接口方法，直接执行代码中的SQL语句，包含业务逻辑

4# 处理执行结果ResultSet

5# 释放资源

JDBC包含臃肿的业务逻辑代码

并且直接将SQL语句写在代码块中，使代码都耦合在一起

1. **JDBCtemplate**

Spring将JDBC封装整合，为其提供了Spring的注入功能，并将JDBC的数据库操作与业务逻辑分离，将事务控制、资源管理、异常处理等事务封装进了Spring框架里。因此JDBCtemplate只进行数据访问逻辑

区别：

* 整合JDBC业务逻辑进Spring框架
* 将数据库连接等配置作为bean，写入bean配置文件内，直接调用
* 将包含数据库操作的实体类作为bean，写入bean配置文件内，直接调用

但是数据库的具体操作和调用方法耦合在一起，不方便修改

1. **Mybatis**

MyBatis是一款优秀的持久层框架，它支持定制化SQL、存储过程及高级映射。将Mybatis整合进Spring框架，结合了Spring框架的事务管理，并结合自身的高级映射特性，方便简化了底层的操作。

区别：

* 使用XML和注解来配置和映射原生类型、接口、实体类
* 将数据库连接等配置作为bean，写入bean配置文件内，直接调用
* 将数据库操作配置进映射文件，分离了接口和实体类，降低耦合

1. 接着就是讲Mybatis整合进Spring框架，分离DAO层和服务层，用接口进行连接
2. 然后购买服务器，在服务器上配置java和maven环境，然后将本地Spring项目打包成jar包发送给服务器，在服务器上运行main方法
3. 最后进行数据库的增删改查的批量操作，检测数据库操作的执行效率和问题
4. **深度思考**
5. **Mybatis有哪些常用标签？怎么使用标签来完成动态查询？**

**Mybatis常用标签有**：



**Mybatis动态标签有**：

* if

标签通常用于WHERE语句、UPDATE语句、INSERT语句中

通过判断参数值来判断：

* 是否使用某个查询条件
* 是否更新某一个字段
* 是否插入某个字段的值
* choose

搭配when属性使用，按顺序判断when中的条件是否成立，成立则结束

* foreach

标签能将list，map等集合遍历插入SQL语句，多用于进行大批量插入

标签主要用于构建in条件，可在sql中对集合进行迭代

属性介绍：

* collection：

属性的值分别是list、array、map三种

对应参数类型为List、数组、map集合

* item：表示在迭代过程中每一个元素的别名
* index：表示在迭代过程中每次迭代到的位置（下标）
* open：前缀
* close：后缀
* separator：分隔符，表示迭代时每个元素之间以什么分隔
* set

标签可以动态配置set关键字，剔除追加到条件末尾的任何不相关的逗号

在update语句中使用if标签时，如果if没有执行，则或导致逗号多余错误

* where

标签添加限定条件

* trim

标签多用于拼接sql片段，可以加前缀后缀，去除sql片段前面或后面的and、or或“，”

1. **什么叫反射？反射的坏处是什么？有哪些反射的应用场景？**

顾名思义，在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能动态地获取该对象的信息，以及动态地调用该对象，这种方法叫做java的反射机制。

反射提高了程序的灵活性和扩展性，降低耦合性，提高自适应能力。它允许程序创和控制任何类的对象，无需提前硬编码目标类。

反射的缺点是性能问题，使用反射基本上是一种解释操作，用于字段和方法接入时要远慢于直接代码。如我们给软件安装插件就是一种反射。

1. **什么是SVN，小乌龟是什么，SVN的文件版本号是怎么来的，哪些文件应该上传到SVN，哪些不该上传？Git和SVN的区别又是什么？**
2. SVN是集中式版本控制系统，小乌龟是SVN的客户端
3. Git是分布式版本控制系统
4. 该上传src文件夹，及pom文件
5. **什么是AOP，适用于哪些场景，AOP的实现方案有哪些？**

AOP称为是面向切面编程，那么对它最好的解释就是拦截器

AOP的原理为：

* 在执行某些代码之前执行另外的代码，使程序变的灵活，扩展性更灵活，可以随意的删除和添加某些功能。适用于打日志，权限管理，异常处理等场景

实现方案有三种：

* 利用代理模式动态的实现AOP
* 使用预编译的方法静态进行代理
* 使用自定义加载器的方法动态进行代理

1. **为什么要使用Interface，而不是直接使用一个实体类来完成任务？Interface和Impl这种方式的好处是什么？**

首先接口是个规范，接口进行了抽象，实现多态

其次直接使用实体类代码耦合在一起，后期做代码修改时比较难改

Interface和Impl分离可以降低代码耦合度

1. **日志应该怎么打，在什么位置，需要打印出来什么样的关键参数？**

可以通过system.out.println（）和log4j打印到控制台，log4j还可以打印到指定.log文件，需要打印出输出级别，线程，时间，代码行数

1. **Spring中的IOC是什么意思，为什么要用IOC而不是New来创建实例？**

IOC容器：就是具有依赖注入功能的容器，是可以创建对象的容器，IOC容器负责实例化、定位、配置应用程序中的对象及建立这些对象间的依赖。

DI(依赖注入Dependency injection) ：在容器创建对象后，处理对象的依赖关系

通常new一个实例，控制权由程序员控制，而"控制反转”是指new实例工作不由程序员来做而是交给Spring容器来做，这样减少了程序员的操作，降低工作量，减少工作失误。

1. **什么是贫血模型，什么是充血模型？为什么我们会强制要求使用贫血模型？**

贫血模型：指Model 中，仅包含状态(属性），不包含行为(方法），采用这种设计时，需要分离出DB层，专门用于数据库操作

充血模型：指Model 中既包括状态，又包括行为，是最符合面向对象的设计方式

因为贫血模型系统的层次结构清楚，各层之间单向依赖，适合初学者，充血模型用不好的话就是两种模型混用，纯粹是为了面对对象而面对对象

1. **怎么进行分页数据的查询，如何判断是否有下一页？**

在开发过程当中，接触比较多的有MySQL、SQLServer、Oracle这三种数据库，这三种数据库都有自己内部的方法可以进行分页查询

* MySQL用的是limit进行分页查询
* SQLServer用的是row\_number函数
* Oracle用的是ROWNUM、row\_number()

1. **MySQL的limit用法**

select \* from tablename limit #{start},#{size}

start是从哪里开始显示

size是每页显示多少条记录

**明天完成的事情：**

明天开始接触任务二的内容

**遇到的问题：**

今天没有进行代码编程，基本进行任务一的知识回顾，没有遇到问题

**收获：**

重新回顾了一遍任务结构，梳理了知识网络，对学到的知识有了个更清楚的认识