读《Java EE互联网轻量级框架整合开发 SSM框架（）Sring MVC+Spring+Mybatis 及redis的实现》

第一部分：整体上了解

Spring框架是java应用最广的框架。其拥有优秀的理念，包括IoC（Inversion of Control，控制反转）和AOP（Aspect Oriented Programming，面向切面编程）

1.1Spring IoC简介

IoC是一个认为一切java资源都是Java Bean的容器，其目标就是装载和管理这些Bean和Bean之间的关系。Bean之间的关系，如老师和学生组成班级，同时老师Bean和学生Bean明显存在教育与被教育的依赖关系。这些都不是人工完成的，而是Spring IoC通过描述自己自动创建的。

即——Spring通过描述，完成对对象的创建和管理其依赖关系。

描述即是指的XML配置文件。如，原来配置两个插座代码是：

Socket socket = new Socket1();

User.setSocket(socket);

User.useSocket();

Socket socket = new Socket2();

User.setSocket(socket);

User.useSocket();

通过两端代码可以看出，如果用户想继续增加插座，就需要修改源代码增加插座n，可是，互联网用户千千万万，如果每一次都要修改源码，岂不是不可能。所以，IoC通过XML描述配置文件取代以前那种笨重的方式。

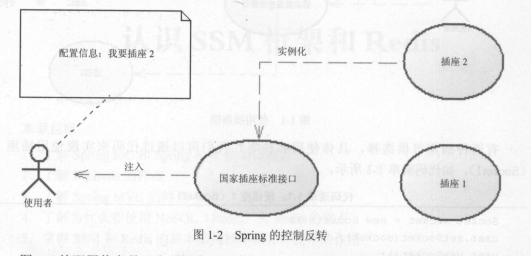
<bean id = “socket” class = “Socket1”/>

//<bean id = “socket” class = “Socket2”> 如果想增加插座数量，只需增加这么一行描述，而不需要修改源码，IoC即可根据描述进行修改。

<bean id = “user” class = “xxx.User”>

<property name = “socket” ref = “socket”/>

</bean>



上图中的配置信息“我要插座2”，相当于XML依赖关系的配置。IoC容器通过描述信息被动的增添“插座2”，但是增添之后，又能拥有“插座2”的主导权，这就是控制反转的定义。

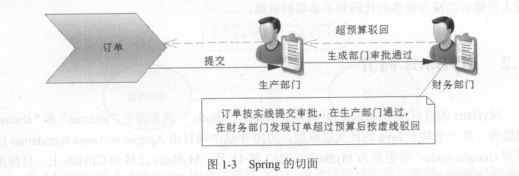
总结：使功能接口不再依赖某一个类，也不需要使用关键字new，而是通过控制信息“即插即拔”的管理他们之间的关系。通过Spring IoC将Bean之间的依赖关系解耦，并且IoC还提供对Java Bean生命周期的管理功能，可以延迟加载，可以在其生命周期内定义一些定义。

1.2 Spring AOP

Bean使java面向对象的设计，IoC对其进行管理即是面向对象（OOP）的处理。

那么非面向对象的处理怎么办？

举例：生产部门的订单、生产部门、财务部门三者符合OOP的设计理念。可是假使某一天，财务部门发现某一项订单的价格超支了，需要取消订单，还需要将生产部门之前所做的审批全部作废。我们不妨将预算超支这个条件称之为切面。显然，它同时影响了订单、财务部门、生产部门等多个对象，还影响了多个对象之间的协作。所以只用OOP是不完善的，还需要面向切面的编程，通过它去管理再切面上的某些对象之间的协作。



实线使提交订单的流程，虚线是一个订单驳回的流程，使切面条件：预算超额，影响了它。

切面条件-数据库更新异常：我们知道，只要Spring接收到了异常，就会令数据库进行回滚，从而保证数据的一致性。

Spring AOP作用使替代代码实现发现异常进行回滚的功能。



说白了，就是对回滚功能进行封装，使代码可以无限次重用。

第一块Spring总结：有了Spring的这些封装，开发人员可以减少很多的代码和不必要的麻烦。

2.MyBatis简介

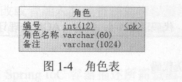
MyBatis前身使Apache的一个开源项目iBatis（“internet”和“abatis“的组合），是一个基于java的持久层框架。2010年这个项目由”Apache software foundation“迁移到Google code，并更名为MyBatis，2013年，此项目迁移到Github上，并由github维护至今。

MyBatis的优势在于灵活，它几乎可以代替JDBC，同时提供了接口编程。目前MyBatis的数据访问层DAO（Data Access Objects）是不需要实现类的，它只需要一个接口和XML（描述注解）。MyBatis提供自动映射、动态SQL、级联、缓存、注解、代码和SQL分离等特性，使用方便，同时也可以对SQL进行优化。因为其具有封装少、映射多样化、支持存储过程、可以进行SQL优化等特点，使得它取代了Hibernate成为了Java互联网中首选的持久框架。

Hibernates作为一个同样十分流行的框架，必然同样也有着MyBatis轻易无法取代的优势，

2.1 Mybatis和hibernate的区别：

二者都是持久层框架，都会设计数据库，所以首先要定义一个数据库表——角色表（t-role），其结构如下图所示：



根据这个角色表，我们可以用一个POJO（Plain Ordinary Java Object）和这张表定义的字段对应起来。

Package com.learn.chapter1.pojo;

Public class Role implements java,io,Serializable{

private Integer id;

private String roleName;

private String note;

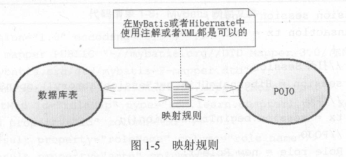
/\*\* setter and getter \*\*/

}

无论是Mybatis还是hibernate都是依靠某种方法，将数据库的表和POJO映射起来的，这样程序员就可以操作POJO来完成相关的逻辑了。

2.2 Hibernate简介 （不需关注SQL，业务逻辑更加清晰）

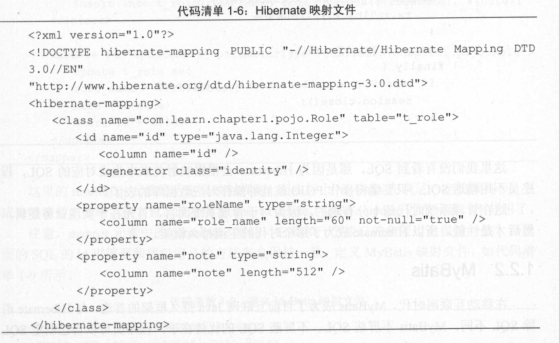
要将POJO和数据库映射起来需要给这些框架提供映射规则，所以要如何提供映射规则呢？



在MyBatis或者Hibernate中可以通过XML或者注解提供映射规则，这里讨论的使XML方式，因为在MyBatis中注解方式会受到一定的限制，所以MyBatis通常使用XML方式实现映射关系。

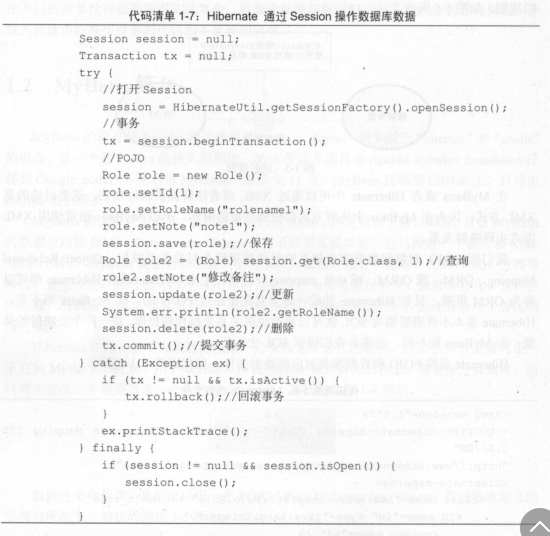
（能通过XML的尽量通过XML）

介绍一个概念：对象关系映射（Object Relational Mapping，O/RM，或O/R mapping）框架，定义即是POJO对象和数据库表相互映射的框架。无论是Mybatis后者Hibernate都可以称为ORM框架。但是又有所不同：Hibernate是一个完全的ORM框架，而MyBatis是一款不完全ORM框架。因为Hibernate的设计理念是完全面向POJO的，而MyBatis则不是。Hibernate基本不在需要编写SQl就可以通过映射关系来操作数据库，是一种全表映射的体现；而MyBatis仍需要在一定程度上依赖SQL去运行，是一种非全表映射。



Hibernate将POJO和数据库对应的映射文件

首先，对POJO和表t-role进行了映射配置，把两者映射起来了。然后，对POJO进行操作，从而影响t-role表的数据，比如对其增、删、查、改就可以按照以下进行操作：



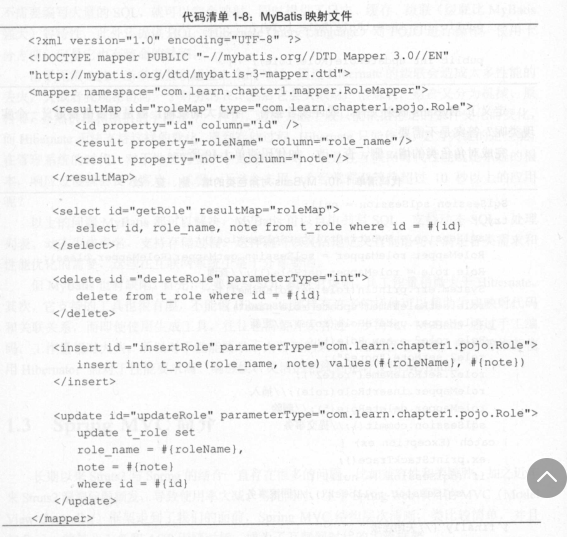
所以，利用Hibernate框架，程序员不需要精通SQL，只要懂得操作POJO就能够操作对应数据库的表了。

对于管理系统而言，第一重要的是实现业务逻辑，然后才是系统的性能，所以Hibernate框架正好适合开发管理系统，所以曾经成为了主流的框架。

2.3 MyBatis（性能好）

MyBatis不屏蔽SQL，自然就有不屏蔽SQL的优势：程序员自己制定SQL规则，无需Hibernate自动生成规则，这样可以更加精确地定义SQL，从而优化性能。

同Hibernate一样，MyBatis也需要将一个映射文件和数据库地表对应起来。MyBatis映射文件如代码清单所示：



这里的resultMap用于定义映射规则，而实际上MyBatis在满足一定的规则下，使用insert、delete、select、update四个元素来完成增删改查四个功能。

值得注意的使：mapper元素中的namespace属性要与一个接口的全限定名保持一致，而里面的SQL的id也需要和接口定义的方法保持一致，定义MyBatis映射文件。

Public com.learn.chapter.mapper;

Import com.learn.chapter.pojo.Role;

Public interface RoleMapper{

Public Role getRole(Integer id);

Public int deleteRole(Integer id);

Public int insertRole(Role role);

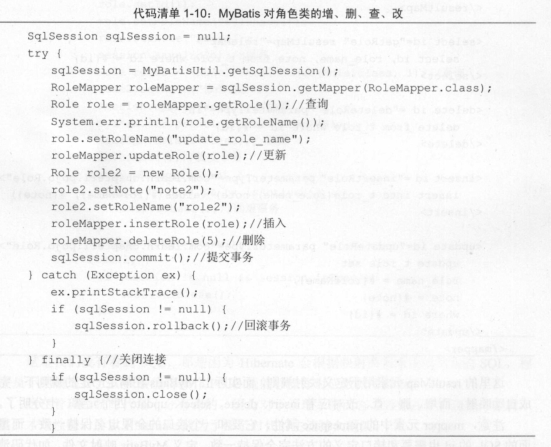
Public int updateRole(Role role);

}

定义M有B按提示映射文件后，就会有一个问题，是否需要定义一个实现类呢？

答案使不需要。

完成对角色类的增删改查，代码如下：



显然MyBatis在业务逻辑上是和Hibernate大同小异的。其区别在于：

MyBatis：提供接口和SQL的利弊： 利：有很高的灵活性，如自定义SQL、映射关系，灵活性和可优化性超过Hibernate。 弊：工作量比Hibernate大。

又有一个问题，为何SQL的灵活性和可优化性十分重要呢？

因为实现同一功能的一条SQL的性能可能相差十几倍到几十倍，这对性能影响非常

之大

总结： MyBatis：性能好，业务逻辑比较复杂，需要自己定义SQL，因为实现的不是完全映射，保留下来了自己优化的空间。

Hibernate：业务逻辑清晰，不需掌握SQL语句，也就是说实现了完全映射，同时提供了日志、缓存、级联（级联比MyBatis强大）等特性，此外还提供HQL（Hibernate Query Language）对POJO进行操作，使用十分方便，但同时也带来了致命缺陷。

Hibernate致命缺陷：由于无需SQL，当夺标关联超过个时，通过Hibernate的级联会造成太多性能的丢失，在互联网时代系统性能很重要，试想谁会愿意去使用一个经常需要等待秒以上的应用呢？

MyBatis也有一定的缺陷：因需要程序员自己编写SQL和映射规则，工作量稍大。

所以，对于系统性能要求并不严苛的系统，比如管理系统、ERP等推荐使用Hibernate

对于系统性能要求高、要求响应快、灵活的系统则推荐使用MyBatis。

3.Spring MVC简介

产生的背景：Struts2框架与Spring的结合一直存在很多的问题，比如兼容性和类臃肿。加之近年来Struts2的漏洞问题频发，导致使用率大减。

从Spring Web项目中产生的MVC（Model View Controller）框架，层次清晰，类比较简单，并且与Spring的核心IoC和AOP无缝对接，成为了互联网时代的主流框架。

MVC模式把应用程序（输入逻辑、业务逻辑和UI逻辑）分成不同的方面，同时提供这些元素之间的松耦合。

Model（模型）：封装了应用程序的数据和由他们组成的POJO。

View（视图）：负责把模型数据渲染到视图上，将数据以一定的形式展现给用户。

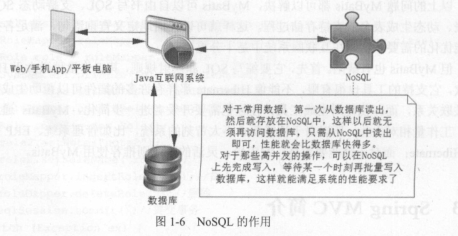
Controller（控制器）：负责处理用户请求，并建立适当的模型把它传递给视图渲染。

在Spring MVC中还可以定义逻辑视图，通过其提供的视图解析器就能够很方便的找到对应的视图进行渲染，或者使用其消息转换的功能，比如在Controller的方法内加入注解@ResponseBody后，SpringMVC就可以通过其消息转环系统，将数据转换为JSON，提供给前端Ajax请求使用。

Spring MVC中的重点在于它的流程和一些重要的主角，包括控制器、视图解析器、视图等重要内容。

3.6最流行的NOSQL——Redis

Redis是当前互联网世界最为流行的NoSQL（Not Only SQL）。NoSQL在互联网系统中的作用很大，因为它可以在很大程序上提高互联网系统的性能。它具备一定持久层的功能，也可以作为一种缓存工具。对于NoSQL数据库而言，作为持久层，它存储的数据是半结构化的，这就意味着计算机在读入内存中由更少的规则，读入速度更快。对于哪些结构化、多范式规则的数据库系统而言，它更具性能优势。作为缓存，它可以支持大数据存入内存中，只要命中率高，它就能快速的相应，因为在内存中的数据读/写比数据库的读/写磁盘的速度快几十到上百倍如下图所示：



，

目前NoSQL有很多争议，有些人认为它可以取代数据库，但是还有一种观点，因为数据库系统有更好的规范性和数据完整性，功能更强大，作为持久层更为完善，安全性更高。而NoSQL结构松散、不完整，功能有限，目前尚不具备取代数据库的实力，但是作为缓存工具，它的高性能、高相应等功能，使它成为一个很重要的工具。

当前Redis已经成为了主要的NoSQL工具，其原因如下。

相应快速：Redis响应非常快，每秒可以执行大约110，000个写入系统，或者81，000个读操作，其速度远超数据库。如果存入一些常用的数据，就能有效一高系统的功能。

支持6种数据类型：它们是字符串、哈希结构、列表、集合、可排序集合和基数。比如对于字符串可以存入一些Java基础数据类型，哈希可以存储对象，列表可以存储List对象等。这使得在应用中很容易根据自己的需要选择存储的数据类型，方便开发。对于Redis而言，虽然只有6种数据结构，但是有两大好处：一方面可以满足存储各种数据结构体的需要；另一方面数据类型少，使得规则响应就少，需要的判断和逻辑就少，这样读/写的速度就更快。

操作都是源自的：所有的Redis的操作都是原子的，从而确保当两个客户同时访问Redis服务器时，得到的是更新后的值（最新值）。在需要高并发的场合可以考虑使用Redis的事务，处理一些需要锁的业务。

MultiUtility工具：Redis可以在如缓存、消息传递队列中使用（Redis支持“发布+订阅“的消息模式），在应用程序如Web应用程序回话、网站页面点击数等任何短暂的数据中使用。

正是因为Redis具备这些优点，使得它成为了目前主流的NoSQL技术，在Java互联网中的到了广泛使用。

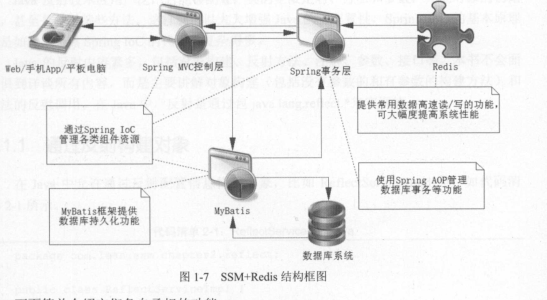
一方面，使用NoSQL从数据库中读取数据进行缓存，就可以从内存中读取数据了，而不像数据库一样读磁盘。现实是读操作远远比写操作多得多，所以缓存中很多常用的数据，提高其命中率有助于整体性能的提高，并且能减缓数据库的压力，对于互联玩系统架构是十分有利的。另一方面，它也能满足互联网高并发需要高速处理数据的场合，如抢红包、商品秒杀等场景，这些场合需要高速处理，并保证并发数据安全和一致性。

总结：NoSQL存储方式简单，存的快，读的快。

数据库：存的更格式化，导致存的慢读的也慢，但是功能更齐全。

3.7 SSM+Redis结构框图及概述

在Java互联网中，以Spring+SpringMVC+MyBatis（SSM）作为主流框架，SSM+Redis的结构框图，如下图所示：



下面简单介绍一下他们各自承担的功能。

Spring IoC：承担资源整合，即插即拔的功能。

Spring AOP可以提供切面管理，特别是数据库事务管理的功能。

MyBatis：提供了一个数据库访问的持久层，通过MyBatis-Spring项目，它便能和Spring无缝对接。

Redis作为缓存工具，它提供了高速度处理数据和缓存数据的功能，使得系统大部分只需要访问缓存，而无须从数据库磁盘中重复读/写；在一些需要高速运算的场合中，也可以先用它来完成计算，再把数据批量存入数据库，这样便能极大地提升互联网系统的性能和响应能力。