

MyBatis: 持久层技术，符合ORM思想的框架技术

那么什么是orm思想呢？

广义上，ORM指的是面向对象的对象模型和关系型数据库的数据结构之间的相互转换。 狭义上，ORM可以被认为是，基于关系型数据库的数据存储，实现一个虚拟的面向对象的数据访问接口。

\*\*O(对象模型)\*\*：实体对象

**R(关系型数据库的数据结构)**：即我们建立的数据库表

**M(映射)**：从R（数据库）到O（对象模型）的映射，可通过XML文件映射。

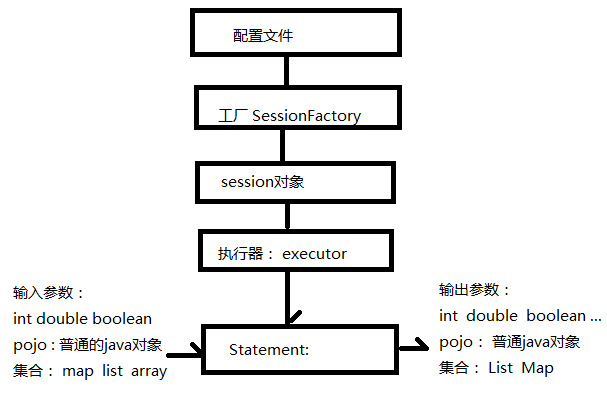
JDBC: 数据模型，和数据库表中的关系没有完成自动化映射关系。

为什么使用mybatis：

Jdbc: 代码冗余, sql和代码耦合不利于程序更新和维护.

Mybatis： 有效的避免了jdbc问题，完成自动化映射，编写灵活sql。

Mybatis执行过程：



Mybatis # $ 区别：

JDBC：Statement(拼接字符串) PreparedStatement(占位符)

#{} 占位符的方式

$ 拼接字符串

Mybatis: 多表联合的方式：

<association property=”关联对象的变量名” column=”关联的字段” javaType=”类型的描述” select=”其他工作空间下的查询标签id”> 关联一个对象

<collection property=”关联对象的变量名” column=”关联的字段” ofType=”类型描述” select=”其他工作空间下的查询标签id” > 关联的是一组对象

动态sql(一个和数据库交互的方法可以适应多种不同业务，减少代码冗余)：

Jdbc： 繁琐的字符串的拼接方式，时刻注意空格 标点 成立条件

where if set trim foreach(批量操作)

Hibernate和Mybatis的区别：

相同点：ORM思想框架实现

不同点：

(1)

Hibernate 是一个全ORM，完全面向对象的操作 session.save() update() delete() get()

Mybati 半ORM : 需要自己去书写我们对应数据库的sql语句。

(2)查询角度

Hibernate 支持丰富查询方式：hql sql qbc qbe 命名查询

Mybatis支持了sql查询

(3)

mybatis比hibernate灵活

Mybatis 支持手动书写和优化sql语句

Hibernate完全由当前的框架去根据数据库自动生成我们的sql

(4)

移植性

Mybatis支持多个数据库操作，切换我们数据源时候不方便，关系型数据中sql是有一些差异

Hibernate: hibernate中没有手动书写sql语句，我们只需要更换hibernate的方言包就可以解决我们的数据库移植问题。

(5)

缓存角度

Hibernate支持丰富类型的缓存机制，自动缓存。

Mybatis 需要我们自己去完成一个配置

难易程度

Mybatis 容易学习 容易使用 写出高质量sql

Hibernate 入门容, 易精通难 SpringData JPA 使用的就是hibernate注解实现

Spring: 综合类型的框架，程序架构的管理，web层技术支持 数据持久层技术的支持 程序部署调用的支持。全家桶，一站式服务，无缝链接。

SpringIOC:

控制反转：IOC原理

1： 没有IOC 我们对象之间的依赖关系是我们用户使用 new关键字进行对象依赖管理

new 耗费内存过程（动态分配内存的行为）

我们依赖关系，不利于后期的扩展和维护(耦合性过高)

2： 我们在编程中各个组件之间存在依赖，我们不需要再我们代码中主动的new依赖对象，而是将我们依赖的对象的名字或者是类型，传递给容器，容器根据我们的需求将我们的依赖对象动态的注入到当我们依赖程序中。

控制： spring提供了一个容器管理我们业务对象

反转：我们提出依赖对象描述，容器从中找寻到我们的对象注入当前运行逻辑中。

生活中例子： 钟表的齿轮之间的依赖,电脑主机可以任意的插拔我们的外部的存储设备，婚介所....

DI: 依赖注入(可以看做ioc例外一个命名)

Set注入

构造器注入

自动装配： 通过注解扫描

SpringAOP：

面向对象编程： 面向对象中我们一个类就是我们一个最小的结构单元

直接在类中嵌入我们扩展应用。

当前的扩展的代码和我们业务代码之间耦合性过高，扩展和维护不方便。

过滤器 监听器 (aop前身)：

外部按照一定api标准构建一个扩展功能代码：

配置文件中描述当前的过滤器和监听器内容。

弊端： 过滤器只能过滤我们的资源文件（servlet jsp http://www....）

监听器只能在我们域对象的管理，特定对象创建 销毁.... 自动的方法执行

面向切面编程：(横切概念)

没有动态代理之前： 创建一个对象 使用该对象调用方法

动态代理： 创建一个对象 ，给当前对象再次创建一个代理对象 代理对象帮助你执行目标方法，（在执行目标方法的同时引入一些自己的行为）

迪丽热巴 经纪人(代理)：热巴(真实对象) 的替身 (接项目 收钱)

面向切面编程几个概念：

切点： 横切多个模块的入口(a开头的方法)。

连接点：符合切点表达式的正在执行的那个目标方法就是连接点,主程序和切面bean进行关联

通知： 外部的目标代码执行的顺序

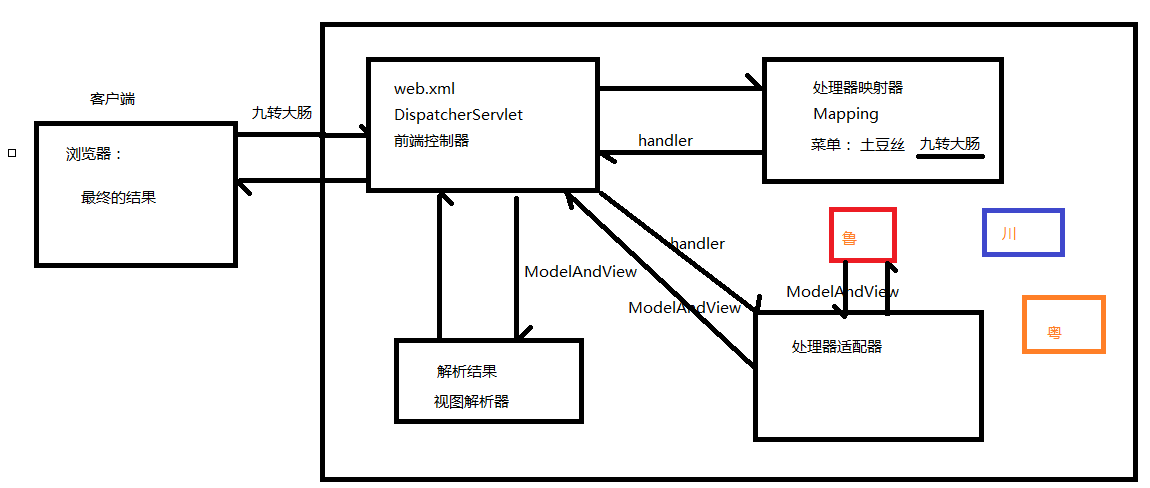
切面： 通知和切点的一个统称

代理： aop实现的过程是采用了代理思想

SpringMVC:

MVC思想的框架，Servlet的一个扩展(SpringMVC 底层设计围绕一个servlet展开)

执行流程：



没有maven:

1: jar包自己手动导入，手动适配

2: 编写--编译--打包--部署--运行。

3：项目内容较多，不易维护更新，扩展，可读性差 难以控制。

Maven：项目管理工具

常用命令： clean test package install

常用指令:

compile : 编译指令 将我们的主程序的java文件编译成我们的class文件

clean : 清除指令 清除掉我们编译和打包后的内容

package: 编译当前项目中的所有的java文同时还将当前项目进行了打件包操作。

install: 在package指令的执行的基础之上，将当前的打包结果放入到本地仓库中

依赖的管理：

scope 依赖的范围: test provided compile

scope作用:

test: 只能是测试文件使用，其他的java代码不能使用,不参与打包 不参与部署和运行(junit)典型的案例

provided: 可以在主程序使用 测试程序中使用,不参与打包不参与部署和运行

(servlet)典型的案例,服务器提供了servlet支持。

compile: 主程序中使用 测试程序中使用，打包 部署 运行

继承：

父子工程：

test provided 不能传递

聚合：

各个子工程聚合成一个完整项目工程，打包 部署 运行

Oracle：

关系型数据库：二维表维护我们对象的关系。

触发器：监听我们对表的一些操作行为

存储过程（数据库编程中一种，多个不同的表进行不同的操作，一次编译，多次使用，提高数据库的执行效率）

序列（设置oracle中一组有效的数字自增的操作机制）

视图（将我们经常使用到数据进行保存，使用视图对象和其他的表进行联合的操作）

索引（表中设定某个字段或者是某些字段设置成索引字段，实现快速查询操作）

函数：oracle中定义的实现一定功能代码块

最重要： 多表的查询程序员必须具备的技能