以前mybatis

先获取session

通过session.insert(“save”,参数)

session.selectOne(“find1”,参数)

session.getMapper()

IUserDao{

public void insert(User user);

}

UserDao implements IUserDao(){

public void insert(User user){

session.insert(“save”, user)

}

}

对这个接口自动生成一个代理对象

*org.mybatis.spring.mapper.MapperScannerConfigurer*

自动去调用这个接口名字所对应的mapper配置文件中的namespace 对应的名字所在的标签id

<mapper namespace=*"day17.* IUserDao *"*>

<insert id=*"* insert*"*>

insert into user(id,name)

values(#{id},#{name})

</insert>

并且把这个代理对象放到spring容器中

所以可以获取这个对象

整合的时候还有一个事务操作

整合的时候，可以不使用spring提供的数据源包，可以用第三方 dbcp 数据源

遵守规范

控制事务，aop控制

生成代理对象

<aop:config

配置

哪个包中的哪个类中的哪个方法使用代理对象

在调用reg方法

之前关闭自动事务提交功能

之后提交事务功能

你可以自己写一个切面来控制这个事务。但是比较繁琐的。

实际开发中，是不需要自己写的。spring帮你做这个事情的，做的很好，很安全

# springmvc

什么是springmvc

也是一种开发模式，mvc模式

模型层，视图层，控制层 分离

就是对这个模式的一个整合。

来一个请求就要配置 一个servlet映射

<servlet>

<servlet-mapping>

100个请求，配置100个映射

而且还要写100个类 doGet() doPost()

对用户的操作

add update find del

就要写4个类

servlet中接受参数

10 参数

request.getParament(“参数名1”)；

request.getParament(“参数名2”)；

request.getParament(“参数名3”)；

request.getParament(“参数名4”)；

request.getParament(“参数名5”)；

进行封装

User user=new User();

user.setId()

user.setId()

user.setId()

user.setId()

等等，确实比较麻烦

mybatis 框架做了很多事情

我们只要写好sql语句 配置文件即可

mvc 遇到这么多问题

springmvc 就是帮我们做那些重复的代码

class UserAction{

add(User user){

自动把前端传过来的数据封装到user对象中

}

del(){

}

update(){

}

}

find(){

}

一个类即可，方便我们维护项目。从这个类中找

等等。

web.xm不需要配置很多映射了，只要一个即可

由这个类分发请求

mybatis中

spring中

都是javaapplication 就可以测试

现在是mvc模式

请求响应

在web服务下运行的

部署在tomcat中

下面就是要新建一个请求类

接受我们的请求

ModelAndView

封装了页面跳转以及数据的封装

视图解决方案

帮助你转发页面

就是为以后维护项目变得更加简单

springmvc 从请求，映射匹配，到handler操作，视图解决方案

变得稍微方便

handler controller 接口，还不是很方便

没有多个方法，以及参数的自动封装

功能越简单，那么性能就越高

配置文件名字为什么是springMVC-servlet.xml

根据servlet名字找这个配置文件的

通过源码 看

请求映射组件 根据url的名字映射的

请求处理组件 controller mutilactioncontroller等

视图组件 可以使用别的

springmvc 如果不配置，那么也会使用默认的方式

spring 运行过程

注解方式

没有返回值，默认返回一个页面 请求的名字的页面