下载jar包 或者用maven引入依赖

设计接口就是设计方法 xml可以用注解代替 对象基本上可以自动生成，参数可以自动传入

自己做好包结构 eclipse有工具 sts 其实spring boot更方便

连接池 引入 然后在spring中配置 pagehelper分页插件

1.dbcp   
2.c3p0   
3.mybatis自带连接池   
4.druid

基础原理

POJO不引用外部任何方法？

Spring 自动地实例化bean 不用new 对象 注解

Springmvc 核心是servlet 拦截请求 匹配控制器调用方法 返回数据或者jsp页面

Mybatis dao层:接口 实现类 调用dao进行业务 数据源

业务层 设计接口 实现

需要引入的依赖 spring框架 以及下属的大量模块 web core aop webmvc(springmvc) oxm tx jdbc

Mysql驱动 mybatis mybatis和spring整合 jstl junit servlet/jsp

Mybatis逆向功能插件

需要

配置 web.xml 载入应用上下文（spring容器） mvc servlet配置 路径 拦截所有url请求 可以禁止访问静态文件 spring配置文件

Mybaits

Mvc 注解映射器 视图解析jsp 路径 后缀

类mapper 实体 服务 控制器 输出的jsp

静态代理 装饰者模式 动态代理 cglib代理

依赖注入 面向接口 pojo 切面 所有的对象都不用new，写好类直接传进去就行

不用在方法内部new外部对象 注入方法 可以用参数法注入到构造方法内，把对象属性赋值为外部对象 这叫构造器注入

应用上下文 xml中写入bean new ApplicationContext(xml).getbean().getA()

切面 日志 安全 事务

写一个类 before after 把对引入实体类 然后执行before after方法 应该用xml声明为切面

Bean工厂（不用了） 应用上下文:从xml中加载配置，从文件系统 或者从类路径（包路径）

一个bean的诞生 实例化 注入属性

Spring 20个模块 aop asm context web jdbc test 主要6个 还有框架

可以用注解减少xml的书写甚至是完全不写xml

可以实现完全的rest 返回json xml 可以利用注解来获取信息如请求头 cookie

声明加载哪些bean 先写接口 带的方法 再继承接口写个bean类

用xml或者注解<beans><aop><mvc><tx><context>

<bean id class > 名字和类的地址（ <constructor-arg value=”>带参构造

注入对象 一样的，先给<bean>加入类 然后在引用的bean中 <constructor-arg ref=”引用对象id”>

如果对象没有构造方法，可以用工厂方法 <bean id class factory-method=”方法名”>

Bean默认是单例，如果想每次生成新的 <bean scope=”prototype”> 作用域:singleton(默认) request每次请求是一个实例 session每次session是一个实例

初始化<bean init-method=””> 销毁destroy-method 可以在beans 中定义默认的初始化和销毁方法

注入属性 <property name value或者ref> 注入空值<null/>

注入内部bean类 <constructor-arg><bean class=></ constructor-arg> 或者也可以夹在<property>中，匿名即可，因为外部永远用不了

可以用命名空间p来装配bean属性 添加声明后 <bean p：name=value p:name-ref=object>

如果属性值是数组或者其他集合时 <property><list> <list>内部子项可以是 <value><bean 内部bean><ref bean=”” 引用外部bean><null><list>

<set>

<map><entry key或key-ref value或value-ref

如果属性值为动态 需要在 <property >value中插入表达式语句 #{} 里面可以是普通值和bean id 可以引用引用对象属性 id.属性名 调用方法 id

T(包名.类名).xxx 访问类静态方法和常量.

运算 比较 逻辑 正则

操作集合 索引访问

三章、最小化xml配置 bean自动装配自动检测

自动装配属性 实际上是自动匹配名字，命名很重要 按名字 按类型(可以自动把相应类实例化装配进来)

Autowire 自动把命名bean装配到同名属性中 <bean autowire >

<beans defalut-autowire》

注解自动装配 开启<context:anotation-config> @Autowire 配合使用qualifier（“id”）注解

@inject

2、完全不需要xml

<context-component-scan>自动检测和定义bean 可以不写<bean> 扫描指定的包和子包

这些bean需要打上注解，必须存在于相应包不然无法识别 @constroller @service等

自动扫描可以配置过滤器

设置@configura的类

四、AOP

通用行为 使用继承和委托的缺点 重复调用

通知 before after(无论方法是否成功 ) after-returning（方法成功后） after-throwing（方法异常后）

切点 切面

连接点 时机

引入 现有类加入新的方法属性

织入

Spring 基于代理 aspectJ 注解 纯pojo 注入式aspectj切面

代理类封装了目标bean类 拦截了方法调用 before后转发再after 基于动态代理

2、使用aspectJ的表达式语言定义切点

切点指示器 arg() execute()this()target()thinin()@arg execution（\* 方法(..)）

Bean指示器

3、使用 xml <aop after before pointcut=”” 传参数

4、增加新方法

5、注解切面 先在<aop:aspect-autoproxy/>就可以自动识别注解 @aspect @pointcut @before

五、数据访问 jdbc hiber dao数据访问对象

1、spring带的数据异常，比jdbc强，和数据框架无关 2、模板化 jdbc-template

3、dao支持类

4、数据源 jdbc jndi 连接池<bean id=”datasource>

5、orm 对象关系映射 模板类 声明事务 dao支持类 上下文session 持久化api

六、事务管理 全有或全无，回滚机制 出了一个错就全部回滚 原子性 一致性 隔离性 持久性（结果存储 不同的框架使用不同的事务管理器 hiber jdbc mybatis

2、声明式事务 事务属性 传播行为 隔离级别 只读 超时 回滚规则

Xml 中定义tx<tx:advice>

3、注解驱动 <tx:annotion-driven/> 完全 注解 @Transactional()注解bean

七、mvc

Restful 和jsp

处理器映射 handlermapping 控制器

请求到Dispatherservlet前端控制器 查询处理器映射 发送给不同控制器

1、web.xml中加入 <servlet> <-name><-class> 设置url的类型比如后缀 <url-pattern>

静态资源<mvc:resources mapping=>s

2、处理映射器 使用注解映射是最方便的 也可以用bean名字 ，类名之类的映射

<mvc: annotion-driven/> @Request-mapping()

3、写控制器类 方法加注解

4、配置视图解析器 添加地址前缀和后缀

5、RequestMapping(value=”” method=””) 类级别设置的是根目录 方法级别会在类的下一层 /a/a

关于查询字符串传参 ?

或者方法（@RequestParam() ） 不用注解其实也可以绑定在同名参数上

关于路径变量value=”/{}” 方法（@PathVariable） 占位符

6、contextloaderlisener 监听所有 xml

7、表单提交

8、返回jsp

9、重定向

10、校验 @valid 注解 @size @pattern

11、文件上传 保存

九、安全 拦截请求 表单认证 权限渲染 记住登录

十、 远程调用 rmi httpinvoke http服务 soap RPC是什么面向服务

十一、REST method @Varable 表单提交 视图解析 json jsp xml @responsebody

首先处理url 基本不直接访问文件 而是关注行为 不用查询参数

Put post get delete curl

@ResponseStatus() 状态码 @RequestMapping(headers={})

资源 json excel pdf html

视图解析器ContentNegotiatingViewResolver

@ResponseBody

2、RestTemplate

十二、JMS

十三、JMX

《spring+mybatis 企业应用实战》

一、领域对象层D0 POJO普通的JAVA对象 DAO层 数据访问 curd数据库

业务逻辑Service 控制器 表现

Servlet jsp JAVABean

Mybatis 把sql配置到配置文件中 或者注解

Model1 model2架构 servlet代替控制器 与javabean（model）通信,model连接数据库

前端控制器dispatcherServlet 配置在web.xml 传入mvc配置文件的路径spring-mvc.xml

<load-on-startup>立即加载 mapping 监听所有url请求。

解析配置文件会产生一个context对象

配置控制器 方法一:mvc.xml中写<bean name=url class=”controller”>

处理器 设置成根据url映射

写控制器类 继承controller接口 必须实现一个handleRequest(res,req)方法 ，必须返回一个ModelAndView 对象 mv.addObject()添加模型 mv.setViewName(“.jsp”)

方法二、注解 不用继承 不用重写方法 不用返回特定

Xml 中要定义组件扫描 注解类型的处理映射器和处理适配器（4.0可以更简单）

3、dispatcherserrvlet 的行为 装配各种解析器 处理映射器 异常解析器 视图名称解析器

配置拦截请求 ，定义控制器

执行控制器 ：请求的json,xml对象转换，将对象转换为json,xml等 数据类型转换

格式化 数字日期 数据验证

三、注解

控制器@controller <context:component-scan>

@request-mapping()value=”url” name method consumes 提交类型 application/json params参数

headers produces 返回类型 注释方法或类

2、处理方法 根据类型可以自动传参

HttpSession session 可以自动传入session ServletRequest response Locale InputStream SessionStatus Errors Model

还可以传入注解 如@RequestParam @RequestHeader @RequestBody @RequestPart

3、关于模型数据 Model ModelMap ModelAndView @ModelAttribute @SessionAttributes

@ModelAttribute方法会先调用

处理方法本身会有一个隐含的model对象，可以传入到处理方法中

Model.addAttribute() 可以添加和取出

4 @RequestParam

5、@PathVariable 和RequestMapping 配合 动态的 地址 可以作为参数传入处理器（不写注解默认传入同名参数）

6、@RequestHeader（value=字段名） 传入请求头

7、@CookieValue(value=”key”) 8、@SessionAttributes 是用于将model存入到Httpsession中的

作用域

8、ModelAttribute 将请求参数绑定到model 5种使用

9、信息转换HttpMessageConvert接口 请求转换为对象 绑定到方法参数 或者输出为响应

实现类 转换为字符串 读取表单 读取xml 读写BufferdImage ByteArray字节数组 json读写

xml配置 httpmessageconvert 请求中指定mime类型

请求头 contenttype multiply-formdata json xml @RequestBody

四。、jsp标签库

隐藏域

五、国际化

加载国际化资源文件 在视图或者控制器输出国际化

六、数据转换

请求数据绑定到处理参数

请求-DataBinder-ConversionService(类型转换，格式化)-入参-validate

String转date

2、格式化 Formatter

3、校验 Validator 方法校验

jsr 303 注解校验

七、上传下载 拦截器 multipart/form-data

上下文配置MultipartResolver 编码，上限等

自动绑定到MuiltipartFile file对象 参数

方法getBytes getContenttype getInputStream getSize isEmpty transferTo()保存文件

2、下载 定义 响应类型 响应头 状态码 ResponseEntity

Return new ResponseEntity(数据，头，码)

八、ORM 和mybatis

Orm 把对对象的操作转变为对数据库的操作

类对应一张表 实例对应一行 列对应类属性

PO 持久化对象 完成curd 只用操作对象就行了

字段 get set 方法

1. mapper.xml <mapper namespace>

<insert id=>语句</insert> 语句中使用表达式#{}来占位

2、mybatis-config.xml <mappers> mapper配置文件 数据源 mysql数据库 事务管理类型

3、读取config.xml 初始化sqlSessionfactory 创建sqlsession

Session方法 .insert(mapper id,被插入的值) .commit()提交事务 关闭session .close

Update delete selectOne selectList selectMap rollback回滚

4、mybatis配置文件的结够

<Select>的属性

resultType=”map” <resultMap>

建议使用Beans或者pojos作为模型 resultType=”包名.模型类” 自动导入

十一、关联映射 多表联查 一对一 多对多

在一个类中引入另一个类 <association property column select=>

1. Mapper接口的代理对象
2. 动态SQL ognl表达式

十二、事务和缓存

配置 创建事务

一级二级缓存

十三、注解 取代xml

@mapper标注dao接口

Select insert xxxxxxprovider result results option one many param

方法命名 selectById

@param配合@insert 直接生成sql语句方法

@Results({@result(property= column=)，@result(property= column=)}) 结果对象属性对应的字段

在controller或者是service中@autowire mapper 注入dao对象 在控制器中 用 @pathvariable 注入路径中的参数变量 方便查询

Return dao.insertA()

注解最大缺点：不用驼峰式就无法映射表字段和实体属性

一对一多对多

动态sql select provider

SQL工具类避免拼接字符串 用来增加子句

参数如何注入到sql的缺省位置 用注

3、用spring-mybatis中间件免除工厂类 可以直接用spring往dao层注入bean

需要 连接池 驱动包

AutoWired 自动注入对象

1. 流程 定义实体类（） 定义 DAO接口（curd操作） 设置注解 设置参数类型 返回类型 动态查询 动态插入

DAO不写实现类，它会自动实现

Service将dao封装在一起 数据访问逻辑 和业务逻辑 定义接口 实现 自动注入dao对象

十四、逆向工具 mybatis generator 代码生成