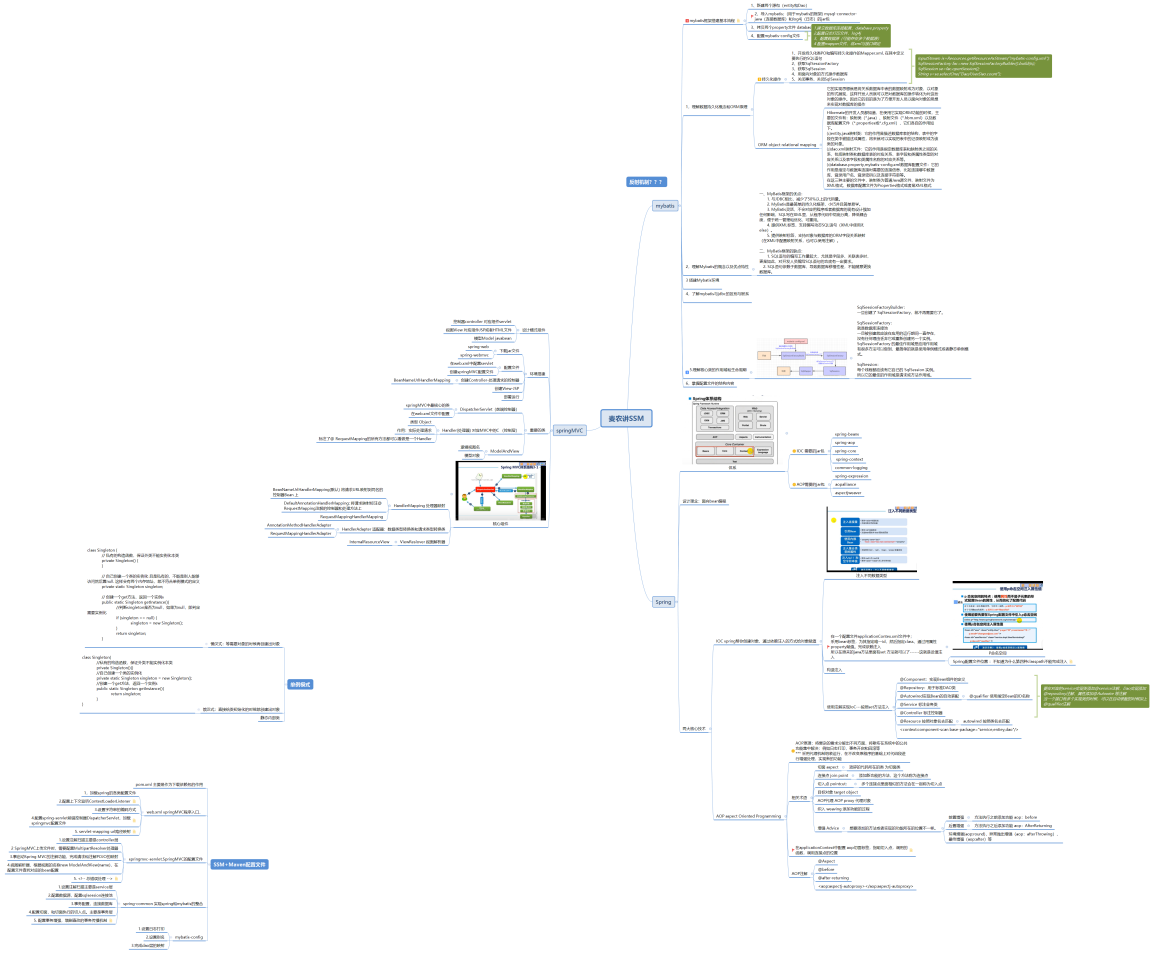
SSM



1. mybatis

1.1. mybatis框架搭建基本流程

1.

1.1.1. 1、新建两个源包（entity和Dao）

1.1.2. 2、导入mybatis；(用于mybatis的框架) mysql-connector-java（连接数据库）和log4j（日志）的jar包

1.1.3. 3、拷贝两个property文件

database（配置数据库名，用户名和密码）和log

1.1.4. 4、配置mybatis-config文件

1.建立数据库连接配置，database.property

2,配置日志打印文件，log4j

3、配置数据源（可能存在多个数据源）

4 配置mapper文件，将xml与接口绑定(4、配置mybatis-config文件)

1.2. 1，理解数据持久化概念和ORM原理

1.2.1. 持久化操作

2

1、开发持久化类PO和编写持久化操作的Mapper.xml,

在其中定义要执行的SQL语句

2、获取SqlSessionFactory

3、获取SqlSession

4、用面向对象的方式操作数据库

5、关闭事务，关闭SqlSession

```
InputStream is=Resources.getResourceAsStream("mybatis-config.xml");
SqlSessionFactory fac=new SqlSessionFactoryBuilder().build(is);
SqlSession se=fac.openSession();
String s=se.selectOne("Dao/UserDao.count");
```

(1、开发持久化类PO和编写持久化操作的Mapper.xml,

在其中定义要执行的SQL语句

2、获取SqlSessionFactory

3、获取SqlSession

4、用面向对象的方式操作数据库

5、关闭事务，关闭SqlSession)

1.2.2. ORM object relational mapping

它的实现思想就是将关系数据库中表的数据映射成为对象，以对象的形式展现，这样开发人员就可以把对数据库的操作转化为对这些对象的操作。因此它的目的是为了方便开发人员以面向对象的思想来实现对数据库的操作

Hibernate的开发人员都知道，在使用它实现ORM功能的时候，主要的文件有：映射类（*.java）、映射文件（*.hbm.xml）以及数据库配置文件（*.properties或*.cfg.xml），它们各自的作用如下。

⑴entity.java映射类：它的作用是描述数据库表的结构，表中的字段在类中被描述成属性，将来就可以实现把表中的记录映射成为该类的对象。

⑵dao.xml映射文件：它的作用是指定数据库表和映射类之间的关系，包括映射类和数据库表的对应关系、表字段和类属性类型的对应关系以及表字段和类属性名称的对应关系等。

⑶database.property,mybatis-config.xml数据库配置文件：它的作用是指定与数据库连接时需要的连接信息，比如连接哪中数据库、登录用户名、登录密码以及连接字符串等。

在这三种主要的文件中，映射类为普通Java源文件、映射文件为XML格式、数据库配置文件为Properties格式或者是XML格式

1.3. 2，理解Mybatis的概念以及优点特性

1.3.1. 一、MyBatis框架的优点：

1. 与JDBC相比，减少了50%以上的代码量。

2. MyBatis是最简单的持久化框架，小巧并且简单易学。

3.

MyBatis灵活，不会对应用程序或者数据库的现有设计强加任何影响，SQL写在

XML里，从程序代码中彻底分离，降低耦合度，便于统一管理和优化，可重用。

4. 提供XML标签，支持编写动态SQL语句（XML中使用if, else）。

5.

提供映射标签，支持对象与数据库的ORM字段关系映射（在XML中配置映射关系，也可以使用注解）。

二、MyBatis框架的缺点：

1.

SQL语句的编写工作量较大，尤其是字段多、关联表多时，更是如此，对开发人员编写SQL语句的功底有一定要求。

2. SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

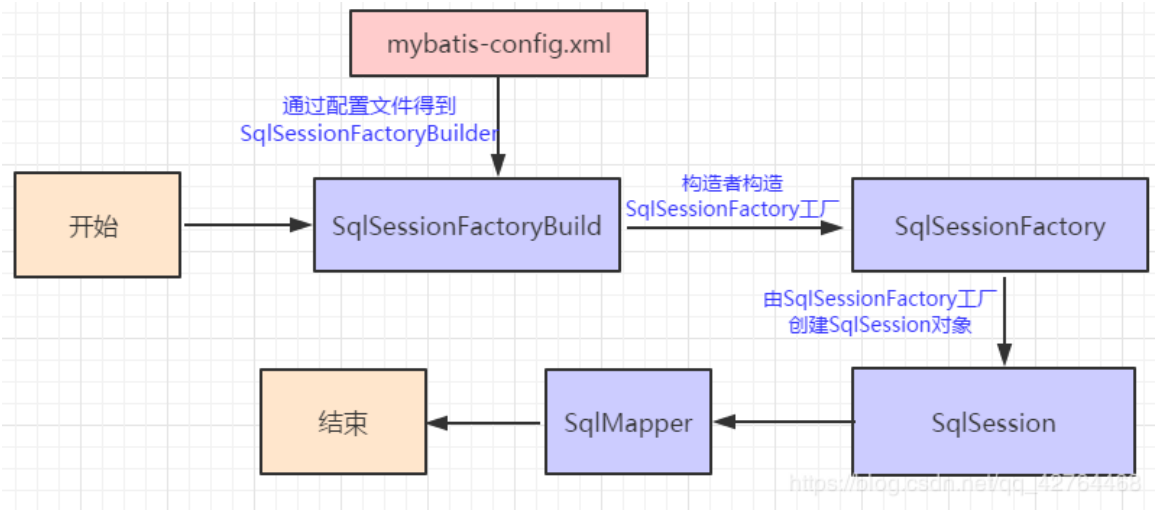
1.4. 3 搭建Mybatis环境

1.5. 4，了解mybatis与jdbc的区别与联系

1.6. 5.理解核心类的作用域和生命周期

3

1.6.1.



SqlSessionFactoryBuilder：

一旦创建了 **SqlSessionFactory**，就不再需要它了。

SqlSessionFactory：

就是数据库连接池

一旦被创建就应该在应用的运行期间一直存在，

没有任何理由丢弃它或重新创建另一个实例。

SqlSessionFactory 的最佳作用域是应用作用域

有很多方法可以做到，最简单的就是使用单例模式或者静态单例模式。

SqlSession：

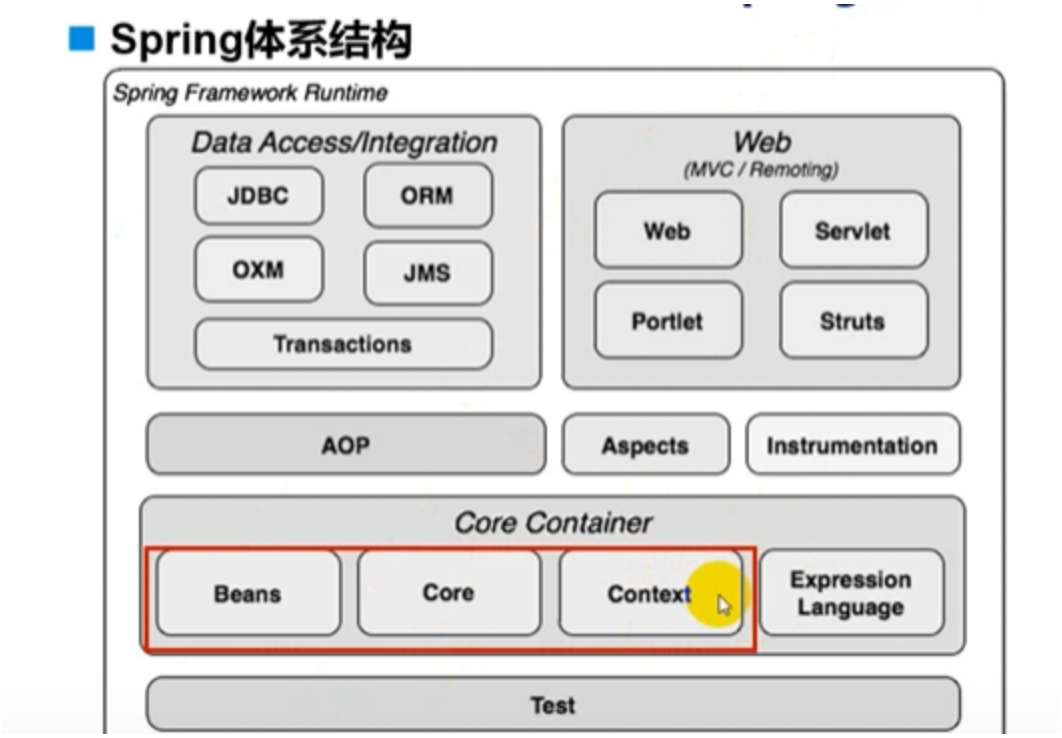
每个线程都应该有它自己的 **SqlSession** 实例。

所以它的最佳的作用域是请求或方法作用域。

1.7.6，掌握配置文件的结构内容

2. Spring

2.1. 体系



2.1.1. IOC 需要的jar包

spring-beans

spring-aop

spring-core

spring-context

common-logging

2.1.2. AOP需要的jar包

spring-expression

aopalliance

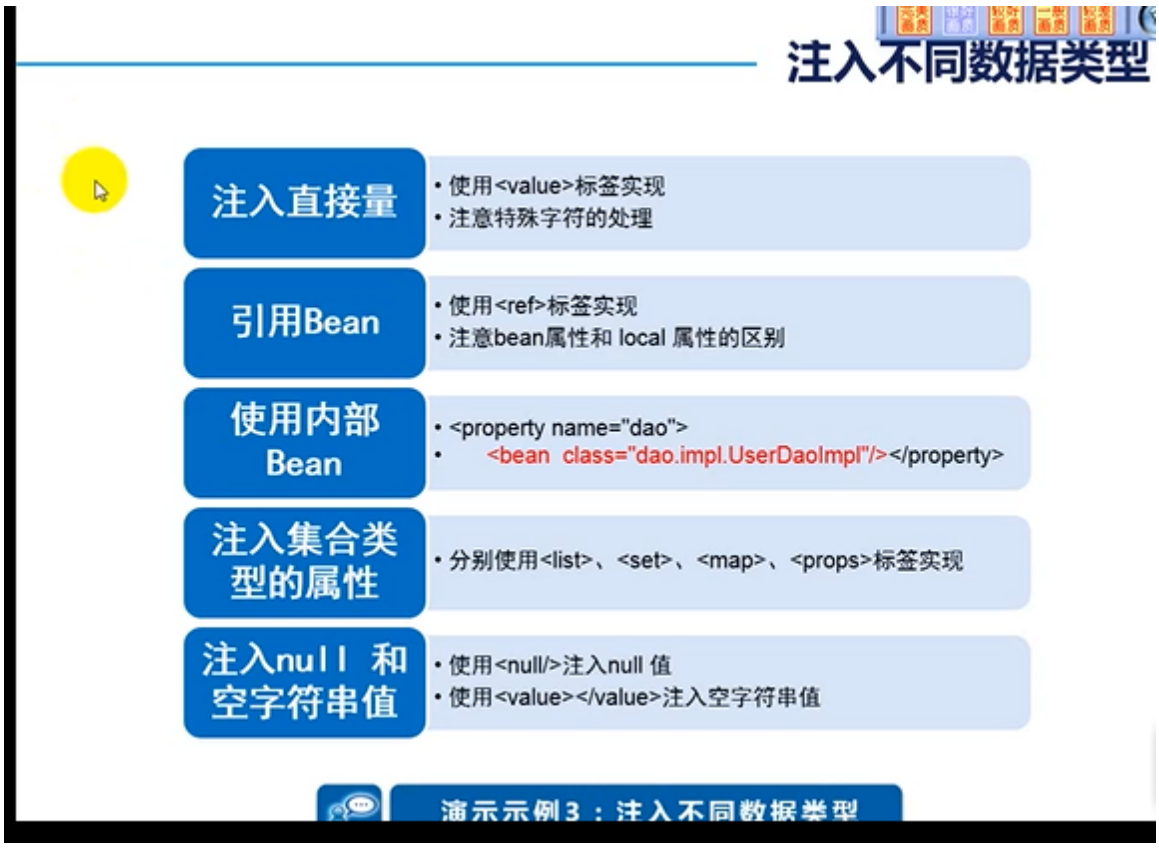
aspectjweaver

2.2. 设计理念：面向bean编程

2.3. 两大核心技术

2.3.1. IOC spring帮你创建对象，通过依赖注入的方式给对象赋值

注入不同数据类型



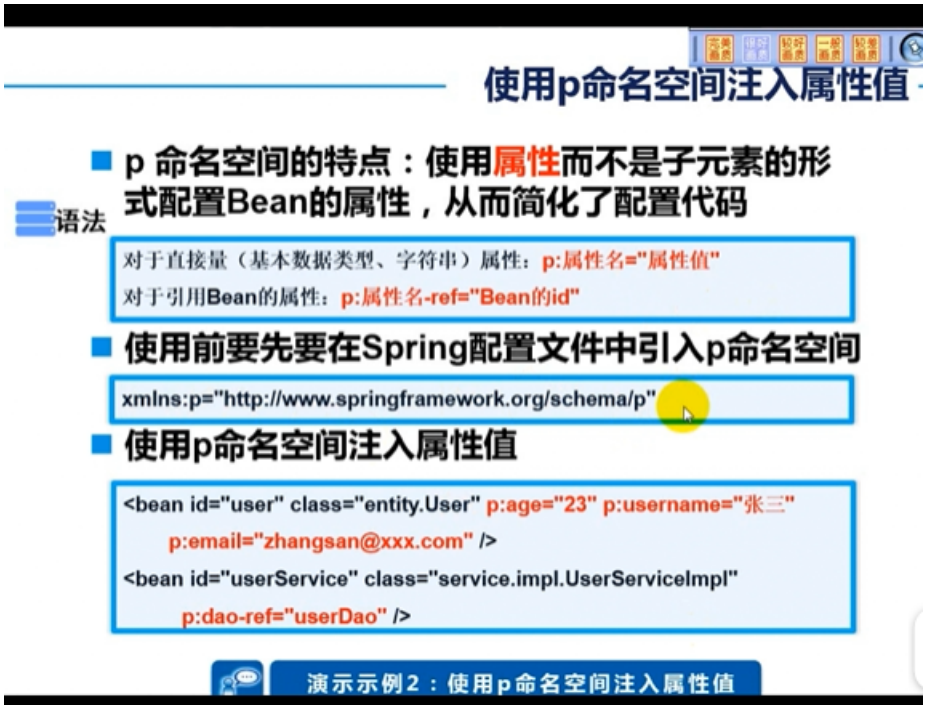
在一个配置文件applicationContex.xml文件中；

采用bean标签，为其指定唯一id，然后指定class、通过用属性property赋值。

完成依赖注入

所以在原来的java方法里面有set 方法就可以了------这就是设值注入

P命名空间



Spring配置文件位置：不知道为什么第四种classpath不能完成注入

spring的配置文件applicationContext.xml的默认地址在WEB-INF下，只要在web.xml中加入代码

org.springframework.web.context.ContextLoaderListener

spring就会被自动加载

但在实际的开发过程中，我们可能需要调整applicationContext.xml的位置，以使程序结构更加的清晰。

在web.xml中，配置Spring配置文件的代码如下：

contextConfigLocation这里写路劲

根据Spring框架的API描述，有以下四种方法配置applicationContext.xml文件路径

1. /WEB-INF/applicationContext.xml
2. com/config/applicationContext.xml
3. file:C:/javacode/springdemo/com/config/applicationContext.xml
4. classpath:com/config/applicationContext.xml

注：以上路径只是举例，具体使用还是要针对真是项目的

开发过程中，如果spring的配置文件applicationContext.xml未加载的话，一般回报这样的错误 Could not open ServletContext resource [/WEB-INF/applicationContext.xml]

下面就有我为大家举例自定义applicationContext.xml路径的常见的方法。

1、Spring配置文件在WEB-INF下面

这种情况你可以不去管他，不进行配置，因为spring会默认去加载，如果一定要配置呢，可以这样

WEB-INF/applicationContext.xml

2、Spring配置文件在WEB-INF下的某个文件夹下，比如config下，可以这样配置

WEB-INF/config/applicationContext.xml

3、Spring配置文件在src下面，可以这样配置

WEB-INF/classes/applicationContext.xml

或者

contextConfigLocation classpath:applicationContext.xml

4、Spring配置文件在src下的某个包里，比如com.config，可以这样配置

WEB-INF/classes/com/config/applicationContext.xml

或者

classpath:com/config/applicationContext.xml

构造注入

使用注解实现IoC---按照set方法注入

@Component：实现Bean组件的定义

@Repository：用于标准DAO类

@Autowired实现Bean的自动装配

@qualifier 使用指定Bean的ID名称

@Service 标注业务类

@Controller 标注控制器

@Resource 按照对象名去匹配

autowired 按照类名去匹配

<context:component-scan base-package="service,entiey,dao"/>

要在对应的service实现类添加@service注解，Dao实现添加@repository注解，属性添加@Autowire 等注解

当一个接口有多个实现类的时候，可以在自动装配的时候加上@qualifier注解(@Component：实现Bean组件的定义, @Repository：用于标准DAO类,

@Autowired实现Bean的自动装配, @Service 标注业务类, @Controller 标注控制器)

2.3.2. AOP aspect Oriented Programming

AOP原理：将复杂的需求分解出不同方面，将散布在系统中的公共功能集中解决：例如日志打印，事务开启和回滚等

***

采用代理机制组装运行，在不改变原程序的基础上对代码段进行增强处理，实现新的功能

相关术语

切面 aspect

琐碎的代码所在的类 为切面类

连接点 join point

添加新功能的方法，这个方法称为连接点

切入点 pointcut：

多个连接点里面相似的方法合在一起称为切入点

目标对象 target object

AOP代理 AOP proxy 代理对象

织入 weaving 添加功能的过程

增强 Advice

想要添加的方法或者实现的功能所在的位置不一样。

前置增强

方法执行之前添加功能 aop：before

后置增强

方法执行之后添加功能 aop：AfterReturning

环境增强(aop:round)，异常抛出增强（aop：afterThrowing），最终增强（aop:after）等

在applicationContext中配置

aop切面标签，指明切入点，调用的函数，调用连接点的位置

```
<aop:config>
   <!-- 切入点(连接点）指明要加入切面的方法 -->
    <aop:pointcut expression="execution(public int addNewUser(archiveWeb.seu.entity.UserEntity))" id="pointcut"/>
<!-- 切面 -->
   <aop:aspect ref="aa">
      <!-- 前置增强 -->
       <!-- aop:before 表示位置，method是AOPAspect里面的方法名，切入点是上面的id -->
      <aop:before method="before" pointcut-ref="pointcut"/>
      <!-- 后置增强 -->
      <!-- aop:after 表示位置，method是AOPAspect里面的方法名，切入点是上面的id -->
      <!-- 返回值的名字要和切面类 定义的返回值名字一样 -->
      <aop:after-returning method="after" pointcut-ref="pointcut" returning ="result"/>
   </aop:aspect>
</aop:config>
```

AOP注解

@Aspect

@before

@after-returning

<aop:aspectj-autoproxy></aop:aspectj-autoproxy>

3. springMVC

3.1. 设计模式组件

3.1.1. 控制器controller 对应组件servlet

3.1.2. 视图View 对应组件JSP或者HTML文件

3.1.3. 模型Model javabean

3.2. 环境搭建

3.2.1. 下载jar文件

spring-web

spring-webmvc

3.2.2. 配置文件

在web.xml中配置servlet

创建springMVC配置文件

3.2.3. 创建Controller-处理请求的控制器

BeanNameUrlHandlerMapping

3.2.4. 创建View-JSP

3.2.5. 部署运行

3.3. 重要的类

3.3.1. DispatcherServlet（前端控制器）

springMVC中最核心的类

在web.xml文件中配置

3.3.2. Handler(处理器) 对应MVC中的C （控制层）

类型 Object

作用：实际处理请求

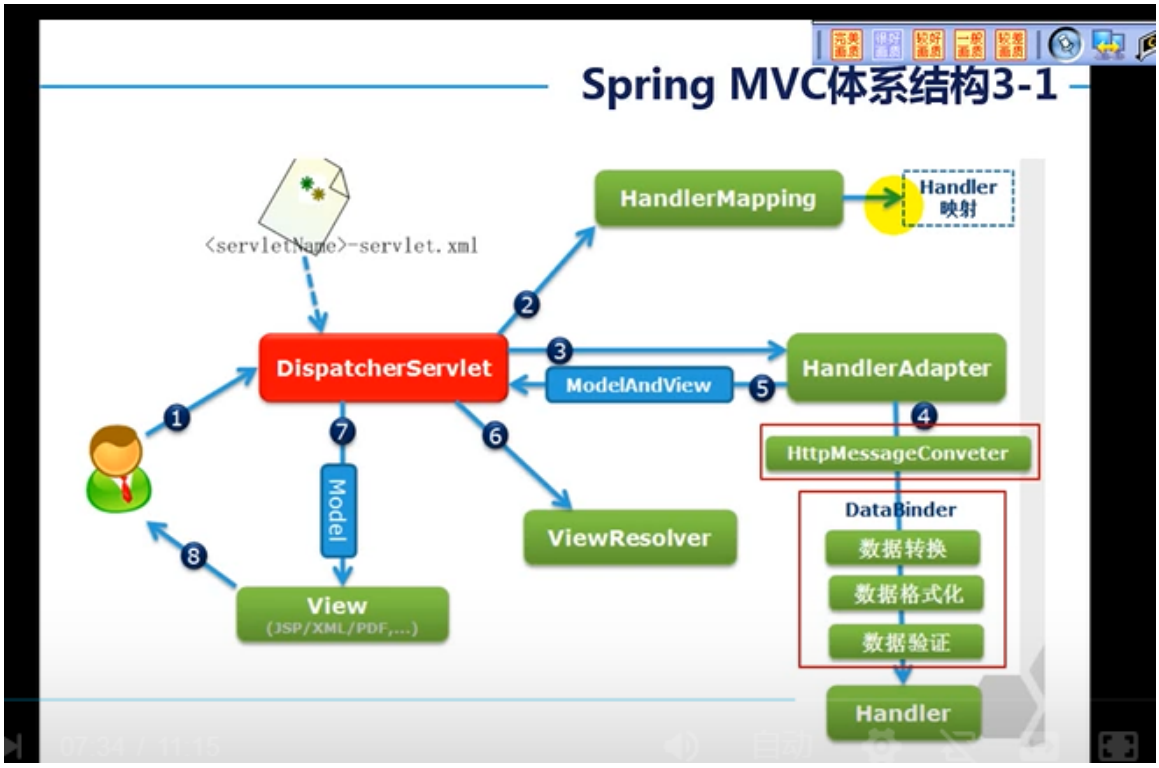
标注了@ RequestMapping的所有方法都可以看做是一个Handler

3.3.3. ModelAndView

逻辑视图名

模型对象

3.4. 核心组件



3.4.1. HandlerMapping 处理器映射

BeanNameUrlHandlerMapping(默认) 将请求URL映射到同名的控制器Bean 上

DefaultAnnotationHandlerMapping: 将请求映射标注@RequestMapping注解的控制器和处理方法上

RequestMappingHandlerMapping

3.4.2. HandlerAdapter 适配器：数据类型转换类和请求类型转换类

AnnotationMethodHandlerAdapter

RequestMappingHandlerAdapter

3.4.3. ViewReslover 视图解析器

InternalResourceView

反射机制？？？

单例模式

懒汉式：等需要对象的时候再创建出对象

```
class Singleton {
  // 私有的构造函数，保证外类不能实例化本类
  private Singleton() {
  }

  // 自己创建一个类的实例化 且是私有的，不能是别人能够访问然后置null.
这样会有两个内存地址，就不符合单例模式的定义
  private static Singleton singleton;

  // 创建一个get方法，返回一个实例s
  public static Singleton getInstance(){
    //判断singleton是否为null，如果为null，即判定需要实例化
    if (singleton == null) {
      singleton = new Singleton();
    }
    return singleton;
  }
```

饿汉式：直接给类初始化的时候就创建出对象

```
class Singleton{
  //私有的构造函数，保证外类不能实例化本类
  private Singleton(){}
  //自己创建一个类的实例化
```

```
    private static Singleton singleton = new Singleton();
    //创建一个get方法，返回一个实例s
    public static Singleton getInstance(){
      return singleton;
    }
  }
```

静态内部类

SSM+Maven配置文件

pom.xml 主要是作为下载依赖包的作用

web.xml springMVC程序入口，

1，加载spring的各类配置文件

2.配置上下文监听ContextLoaderListener

ContextLoaderListener的作用就是启动Web容器时，读取在contextConfigLocation中定义的xml文件，自动装配ApplicationContext的配置信息，并产生WebApplicationContext对象，然后将这个对象放置在ServletContext的属性里，这样我们只要得到Servlet就可以得到WebApplicationContext对象，并利用这个对象访问spring容器管理的bean

3.设置字符串的编码方式

4.配置spring-servlet前端控制器DispatcherServlet，加载springmvc配置文件

DispatcherServlet是前端控制器设计模式的实现，提供Spring Web MVC的集中访问点，而且负责职责的分派，而且与Spring IoC容器无缝集成，从而可以获得Spring的所有好处。

5. servlet-mapping url路径映射

如果url－pattern定义的是路径，那么以后所有对这个路径下资源的请求都会由servlet－name中定义的servlet处理；

如果url－pattern定义的是资源格式例如*.do等，那么对于所有符合这种格式的资源的请求都由指定的servlet处理。

springmvc-servlet.SpringMVC的配置文件

1.设置注解扫描主要是controller层

2 SpringMVC上传文件时，需要配置MultipartResolver处理器

3.事启动Spring MVC的注解功能，完成请求和注解POJO的映射

4.视图解析器，根据视图的名称new ModelAndView(name)，在配置文件查找对应的bean配置

5. <!-- 总错误处理 -->

如果url－pattern定义的是路径，那么以后所有对这个路径下资源的请求都会由servlet－name中定义的servlet处理；

如果url－pattern定义的是资源格式例如*.do等，那么对于所有符合这种格式的资源的请求都由指定的servlet处理。

spring-common 实现spring和mybatis的整合

1.设置注解扫描主要是service层

2.配置数据源，配置sqlsession连接池

3.事务配置，连接数据库

4.配置切面，和切面执行的切入点。主要是事务层

5. 配置事务增强，增删查改的事务传播机制

如果url－pattern定义的是路径，那么以后所有对这个路径下资源的请求都会由servlet－name中定义的servlet处理；

如果url－pattern定义的是资源格式例如*.do等，那么对于所有符合这种格式的资源的请求都由指定的servlet处理。

mybatis-config

1.设置日志打印

2.设置别名

3.完成dao层的映射