框架设计

**1、Spring框架概述**

**1.1 什么是Spring**

Spring是一个开源框架，Spring是于2003 年兴起的一个轻量级的Java 开发框架，由Rod Johnson 在其著作Expert One-On-One J2EE Development and Design中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。框架的主要优势之一就是其分层架构，分层架构允许使用者选择使用哪一个组件，同时为 J2EE 应用程序开发提供集成的框架。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。Spring的核心是控制反转（IoC）和面向切面（AOP）。简单来说，Spring是一个分层的JavaSE/EE full-stack(一站式) 轻量级开源框架。

**1.2 Spring的优点**

（1）方便解耦，简化开发 （高内聚低耦合）

Spring就是一个大工厂（容器），可以将所有对象创建和依赖关系维护，交给Spring管理

spring工厂是用于生成bean

（2）AOP编程的支持

Spring提供面向切面编程，可以方便的实现对程序进行权限拦截、运行监控等功能

（3）声明式事务的支持

只需要通过配置就可以完成对事务的管理，而无需手动编程

（4）方便程序的测试

Spring对Junit4支持，可以通过注解方便的测试Spring程序

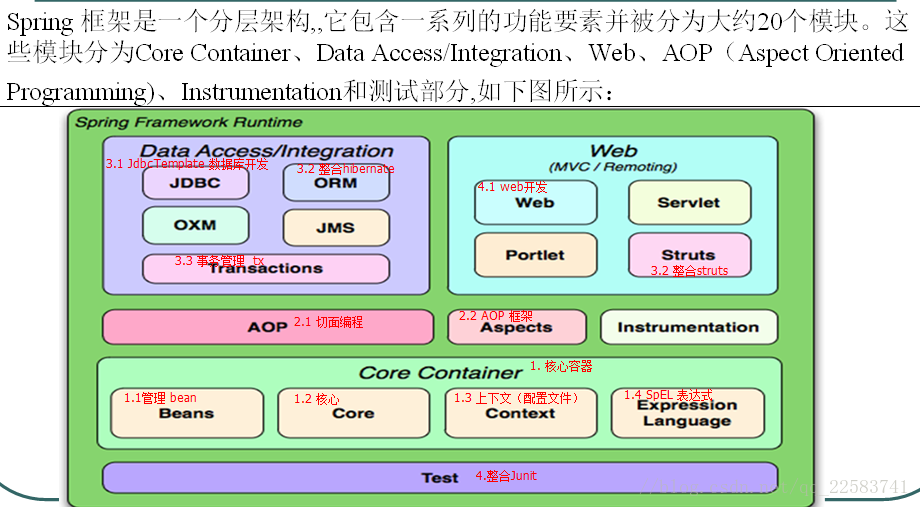
（5）方便集成各种优秀框架

Spring不排斥各种优秀的开源框架，其内部提供了对各种优秀框架（如：Struts、Hibernate、MyBatis、Quartz等）的直接支持

（6）降低JavaEE API的使用难度

Spring 对JavaEE开发中非常难用的一些API（JDBC、JavaMail、远程调用等），都提供了封装，使这些API应用难度大大降低

1.3 Spring的体系结构



Spring框架是由于软件开发的复杂性而创建的。Spring使用的是基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅仅限于服务器端的开发。从简单性、可测试性和松耦合性角度而言，绝大部分Java应用都可以从Spring

◆目的:解决企业应用开发的复杂性

◆功能:使用基本的JavaBean代替EJB，并提供了更多的企业应用功能

◆范围:任何Java应用

Spring是一个轻量级控制反转(IoC)和面向切面(AOP)的容器框架。

### ****2、struts2框架****

Struts2是一个基于MVC设计模式的Web应用框架，它本质上相当于一个servlet，在MVC设计模式中，Struts2作为控制器(Controller)来建立模型与视图的数据交互。Struts2在Struts1融合webwork。ONGL表达式：就是对象图形化导航语言，在前端页面中,访问action的属性、方法;类似于认为就是对JSP的封装,是编程更加方便。

### ****2.1、framework（框架）****

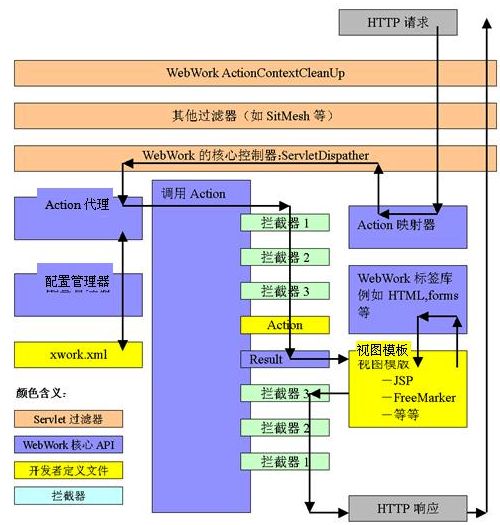
一个基本概念上的结构，用于去解决或者处理复杂的问题，框架是在特定的领域内解决问题。

框架的优点：

(1)重用代码大大增加，软件生产效率和质量也得到了提高

(2)使用框架开发，它提供统一的标准，大大降低了我们的后期维护。

### ****2.2、Struts2框架运作****



工作流程：  
（1）客户端浏览器发送HTTP请求到Web应用  
（2）Web容器将请求传递到标准ActionContextCleanUp过滤器以消除属性，而不让后续过滤器清楚，以延长Action中属性（包括自定义属性）的生命周期。  
（3）再经过如stimesh等其他过滤器后，请求传递给StrutsPrepareAndExecuteFilter核心控制器  
（4）StrutsPrepareAndExecuteFilter调用ActionMapper（Action映射器）确定调用哪个Action，再将控制权转移给ActionProxy代理  
（5）ActionProxy代理调用配置管理器ConfigurationManager从配置文件struts.xml中读取配置信息，然后创建ActionInvocation对象  
（6）ActionInvocation在调用拦截器链中的拦截器后再调用Action，根据Action返回的结果字符串查找对应的Result  
（7）Result调用视图模板，再以相反的顺序执行拦截器链，返回HTTP响应  
（8）HTTP响应以相反的顺序返回给核心控制器StrutsPrepareAndExecuteFilter以及其他web.xml中定义的过滤器，最终返回给客户端。

### **2.3、**Struts2执行流程****

* ****Struts2源码导入（Struts2官网下载地址:**https://struts.apache.org/**）****

 (1) struts2核心部分源代码org.apache.struts2xx : src\core\src\main\java

　　　　(2) struts2的xwork核心部分源代码 :src\xwork-core\src\main\java\com\opensymphony\xwork2

　　　　(3)struts2的插件的源代码: src\plugins

* **关于struts.xml的中文乱码**

　　 第一步：

　　　　在eclipse的window下首选面中查找xml  catalog

　　 第二步：

　　　　Location:配置本地的dtd文件路径

　　　　key type:选择URI

　　　　Key: http://struts.apache.org/dtds/struts-2.3.dtd

　　注意：版本要对应，如果你可以上网，那么会自动缓存dtd,具有提示功能。

* **执行流程介绍**

　　  1.当通过浏览器发送一个请求

　　  2.会被StrutsPrepareAndExecuteFilter拦截

　　  3.会调用struts2框架默认的拦截器(interceptor)完成部分功能

　　  4.在执行Action中操作

　　 5.根据Action中方法的执行结果来选择来跳转页面Resutl视图一般管StrutsPrepareAndExecuteFilter叫做前端控制器(核心控制器)，只有配置了这个filter，Struts2框架才能使　　 　　用，Struts2的默认拦截器(interceptor)它们是在struts-default.xml文件中配置。

3、Hibernate框架

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，它将POJO与数据库表建立映射关系，是一个全自动的orm框架，hibernate可以自动生成SQL语句，自动执行，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的JaveEE架构中取代CMP，完成[数据持久化](https://baike.so.com/doc/5030500-5256785.html" \t "https://baike.so.com/doc/_blank)的重任。

**3.1** **发展历程**

2001年，澳大利亚墨尔本一位名为Gavin King的27岁的程序员，上街买了一本SQL编程的书，他厌倦了实体bean，认为自己可以开发出一个符合对象关系映射理论，并且真正好用的Java持久化层框架，因此他需要先学习一下SQL。这一年的11月，Hibernate的第一个版本发布了。

2002年，已经有人开始关注和使用Hibernate了。

2003年9月，Hibernate开发团队进入JBoss公司，开始全职开发Hibernate，从这个时候开始Hibernate得到了突飞猛进的普及和发展。

2004年，整个Java社区开始从实体bean向Hibernate转移，特别是在Rod Johnson的著作《Expert One-on-One J2EE Development without EJB》出版后，由于这本书以扎实的理论、充分的论据和详实的论述否定了EJB，提出了轻量级敏捷开发理念之后，以Hibernate和Spring为代表的轻量级开源框架开始成为Java世界的主流和事实标准。在2004年Sun领导的J2EE5.0标准制定当中的持久化框架标准正式以Hibernate为蓝本。

2006年，J2EE5.0标准正式发布以后，持久化框架标准Java Persistent API(简称JPA)基本上是参考Hibernate实现的，而Hibernate在3.2版本开始，已经完全兼容JPA标准。

### **3.2** 编程环境

Hibernate是一个以[LGPL](https://baike.so.com/doc/5461694-5700021.html" \t "https://baike.so.com/doc/_blank)(Lesser GNU Public License)许可证形式发布的开源项目。在Hibernate官网上有下载Hibernate包的说明。Hibernate包以源代码或者二进制的形式提供。

**3.3语言特点**

（1）将对数据库的操作转换为对Java对象的操作，从而简化开发。通过修改一个"持久化"对象的属性从而修改数据库表中对应的记录数据。

（2）提供线程和进程两个级别的缓存提升应用程序性能。

（3）有丰富的映射方式将Java对象之间的关系转换为数据库表之间的关系。

（4）屏蔽不同数据库实现之间的差异。在Hibernate中只需要通过"方言"的形式指定当前使用的数据库，就可以根据底层数据库的实际情况生成适合的SQL语句。

（5）非侵入式:Hibernate不要求持久化类实现任何接口或继承任何类，POJO即可。

**3.4核心API**

Hibernate的API一共有6个，分别为:[Session](https://baike.so.com/doc/2366283-2502104.html" \t "https://baike.so.com/doc/_blank)、[SessionFactory](https://baike.so.com/doc/206123-218006.html" \t "https://baike.so.com/doc/_blank)、[Transaction](https://baike.so.com/doc/6951252-7173653.html" \t "https://baike.so.com/doc/_blank)、[Query](https://baike.so.com/doc/6950852-7173253.html" \t "https://baike.so.com/doc/_blank)、[Criteria](https://baike.so.com/doc/6841246-7058527.html" \t "https://baike.so.com/doc/_blank)和Configuration。通过这些接口，可以对持久化对象进行存取、事务控制。

1. Session

Session接口负责执行被持久化对象的[CRUD](https://baike.so.com/doc/6148523-6361708.html" \t "https://baike.so.com/doc/_blank)操作(CRUD的任务是完成与数据库的交流，包含了很多常见的SQL语句)。但需要注意的是[Session对象](https://baike.so.com/doc/2245462-2375809.html" \t "https://baike.so.com/doc/_blank)是非[线程安全](https://baike.so.com/doc/6809842-7026796.html" \t "https://baike.so.com/doc/_blank)的。同时，Hibernate的session不同于JSP应用中的HttpSession。这里当使用session这个术语时，其实指的是Hibernate中的session，而以后会将HttpSession对象称为用户session。

1. SessionFactory

SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建Session对象。这里用到了[工厂模式](https://baike.so.com/doc/6777948-6993946.html" \t "https://baike.so.com/doc/_blank)。需要注意的是SessionFactory并不是轻量级的，因为一般情况下，一个项目通常只需要一个SessionFactory就够，当需要操作多个数据库时，可以为每个数据库指定一个SessionFactory。

1. Transaction

Transaction 接口是一个可选的API，可以选择不使用这个接口，取而代之的是Hibernate 的设计者自己写的底层事务处理代码。 Transaction 接口是对实际事务实现的一个抽象，这些实现包括JDBC的事务、JTA 中的UserTransaction、甚至可以是CORBA 事务。之所以这样设计是能让开发者能够使用一个统一事务的操作界面，使得自己的项目可以在不同的环境和容器之间方便地移植。

1. Query

Query接口让你方便地对数据库及持久对象进行查询，它可以有两种表达方式:HQL语言或本地数据库的SQL语句。Query经常被用来绑定查询参数、限制查询记录数量，并最终执行查询操作。

1. Criteria

Criteria接口与Query接口非常类似，允许创建并执行面向对象的标准化查询。值得注意的是Criteria接口也是轻量级的，它不能在Session之外使用。

1. Configuration

Configuration 类的作用是对Hibernate 进行配置，以及对它进行启动。在Hibernate 的启动过程中，Configuration 类的实例首先定位映射文档的位置，读取这些配置，然后创建一个SessionFactory对象。虽然Configuration 类在整个Hibernate 项目中只扮演着一个很小的角色，但它是启动hibernate 时所遇到的第一个对象。

1. **My Batis 框架**

MyBatis 本是[apache](https://baike.so.com/doc/5333438-5568873.html" \t "https://baike.so.com/doc/_blank)的一个开源项目[iBatis](https://baike.so.com/doc/1381914-1460868.html" \t "https://baike.so.com/doc/_blank), 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。2013年11月迁移到Github。

iBATIS一词来源于"internet"和"abatis"的组合，是一个基于Java的[持久层](https://baike.so.com/doc/7722506-7996601.html" \t "https://baike.so.com/doc/_blank)框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects(DAOs)

当前，最新版本是MyBatis 3.5.0 ，其发布时间是2019年1月20日。

**4.1背景介绍**

MyBatis 是支持普通 SQL查询，[存储过程](https://baike.so.com/doc/2045557-2164405.html" \t "https://baike.so.com/doc/_blank)和高级映射的优秀[持久层](https://baike.so.com/doc/7722506-7996601.html" \t "https://baike.so.com/doc/_blank)框架。MyBatis 消除了几乎所有的[JDBC](https://baike.so.com/doc/2417290-2555569.html" \t "https://baike.so.com/doc/_blank)代码和参数的手工设置以及[结果集](https://baike.so.com/doc/2123416-2246652.html" \t "https://baike.so.com/doc/_blank)的检索。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs(Plain Ordinary Java Objects，普通的 Java对象)映射成数据库中的记录。

每个MyBatis应用程序主要都是使用SqlSessionFactory实例的，一个SqlSessionFactory实例可以通过SqlSessionFactoryBuilder获得。SqlSessionFactoryBuilder可以从一个xml配置文件或者一个预定义的配置类的实例获得。

用xml文件构建SqlSessionFactory实例是非常简单的事情。推荐在这个配置中使用类路径资源(classpath resource)，但你可以使用任何Reader实例，包括用文件路径或file://开头的url创建的实例。MyBatis有一个实用类----Resources，它有很多方法，可以方便地从类路径及其它位置加载资源。

**4.2特点**

（1）简单易学:本身就很小且简单。没有任何第三方依赖，最简单安装只要两个jar文件+配置几个sql映射文件易于学习，易于使用，通过文档和源代码，可以比较完全的掌握它的设计思路和实现。

（2）灵活:mybatis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，便于统一管理和优化。通过sql语句可以满足操作数据库的所有需求。

（3）解除sql与程序代码的耦合:通过提供DAO层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元测试。sql和代码的分离，提高了可维护性。

（4）提供映射标签，支持对象与数据库的orm字段关系映射

（5）提供对象关系映射标签，支持对象关系组建维护

（6）提供xml标签，支持编写动态sql。

**4.3架构功能**

我们把Mybatis的功能架构分为三层:

(1)API接口层:提供给外部使用的接口API，开发人员通过这些本地API来操纵数据库。接口层一接收到调用请求就会调用数据处理层来完成具体的数据处理。

(2)数据处理层:负责具体的SQL查找、SQL解析、SQL执行和执行结果映射处理等。它主要的目的是根据调用的请求完成一次数据库操作。

(3)基础支撑层:负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为最基础的组件。为上层的数据处理层提供最基础的支撑。

4.4框架架构

框架架构讲解:

(1)加载配置:配置来源于两个地方，一处是配置文件，一处是Java代码的注解，将SQL的配置信息加载成为一个个MappedStatement对象(包括了传入参数映射配置、执行的SQL语句、结果映射配置)，存储在内存中。

(2)SQL解析:当API接口层接收到调用请求时，会接收到传入SQL的ID和传入对象(可以是Map、JavaBean或者基本数据类型)，Mybatis会根据SQL的ID找到对应的MappedStatement，然后根据传入参数对象对MappedStatement进行解析，解析后可以得到最终要执行的SQL语句和参数。

(3)SQL执行:将最终得到的SQL和参数拿到数据库进行执行，得到操作数据库的结果。

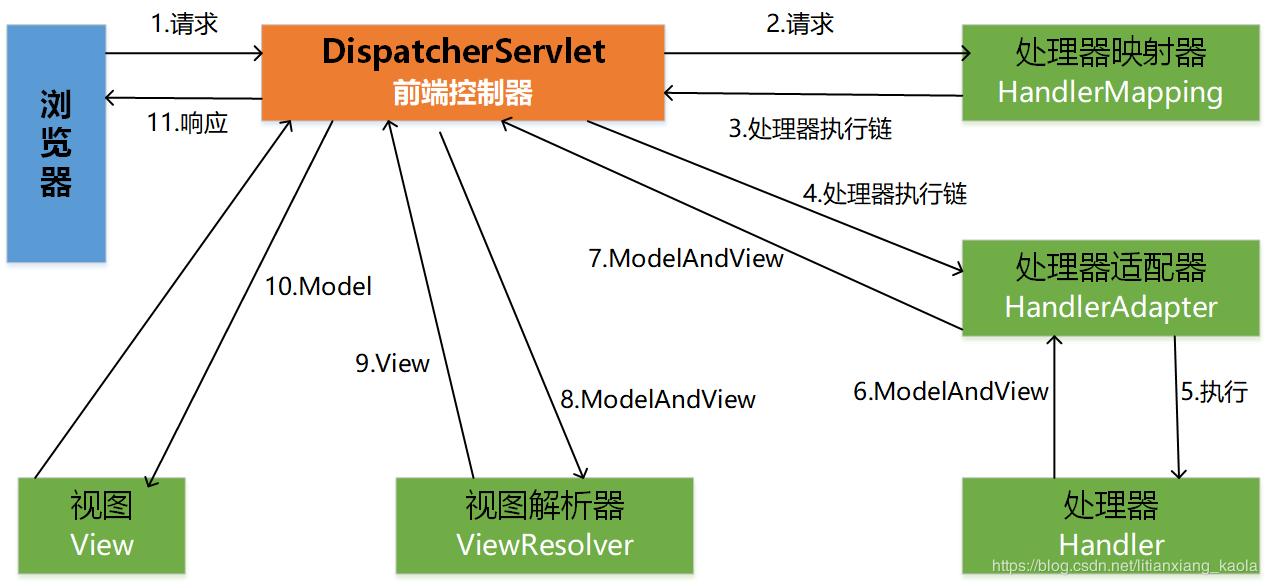
(4)结果映射:将操作数据库的结果按照映射的配置进行转换，可以转换成HashMap、JavaBean或者基本数据类型，并将最终结果返回。

**5.SpringMVC框架**

JavaEE体系结构包括四层，从上到下分别是应用层、Web层、业务层、持久层。Struts和SpringMVC是Web层的框架，Spring是业务层的框架，Hibernate和MyBatis是持久层的框架。

SpringMVC是一种基于Java，实现了Web MVC设计模式，请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将Web层进行职责解耦。基于请求驱动指的就是使用请求-响应模型，框架的目的就是帮助我们简化开发，SpringMVC也是要简化我们日常Web开发。

核心架构



（1）首先浏览器发送请求——>DispatcherServlet，前端控制器收到请求后自己不进行处理，而是委托给其他的解析器进行处理，作为统一访问点，进行全局的流程控制；

（2）DispatcherServlet——>HandlerMapping，处理器映射器将会把请求映射为HandlerExecutionChain对象（包含一个Handler处理器对象、多个HandlerInterceptor拦截器）对象；

（3）DispatcherServlet——>HandlerAdapter，处理器适配器将会把处理器包装为适配器，从而支持多种类型的处理器，即适配器设计模式的应用，从而很容易支持很多类型的处理器；

（4）HandlerAdapter——>调用处理器相应功能处理方法，并返回一个ModelAndView对象（包含模型数据、逻辑视图名）；

（5）ModelAndView对象（Model部分是业务对象返回的模型数据，View部分为逻辑视图名）——> ViewResolver， 视图解析器将把逻辑视图名解析为具体的View；

（6）View——>渲染，View会根据传进来的Model模型数据进行渲染，此处的Model实际是一个Map数据结构；

（7）返回控制权给DispatcherServlet，由DispatcherServlet返回响应给用户，到此一个流程结束。

**5.1 SpringMVC接口解释**

**DispatcherServlet：**

　　Spring提供的前端控制器，所有的请求都有经过它来统一分发。在DispatcherServlet将请求分发给Spring Controller之前，需要借助于Spring提供的HandlerMapping定位到具体的Controller。

**HandlerMapping：**

　　能够完成客户请求到Controller映射。

**Controller：**

　　需要为并发用户处理上述请求，因此实现Controller接口时，必须保证线程安全并且可重用。

　　Controller将处理用户请求，这和Struts Action扮演的角色是一致的。一旦Controller处理完用户请求，则返回ModelAndView对象给DispatcherServlet前端控制器，ModelAndView中包含了模型（Model）和视图（View）。

　　从宏观角度考虑，DispatcherServlet是整个Web应用的控制器；从微观考虑，Controller是单个Http请求处理过程中的控制器，而ModelAndView是Http请求过程中返回的模型（Model）和视图（View）。

**ViewResolver：**

　　Spring提供的视图解析器（ViewResolver）在Web应用中查找View对象，从而将相应结果渲染给客户。

**5.2、读取配置**

SpringMVC本质上是一个Servlet,这个 Servlet 继承自 HttpServlet。

FrameworkServlet负责初始化SpringMVC的容器，并将Spring容器设置为父容器。因为本文只是实现SpringMVC，对于Spring容器不做过多讲解（有兴趣同学可以看看博主另一篇文章：向spring大佬低头--大量源码流出解析）。

为了读取web.xml中的配置，我们用到ServletConfig这个类，它代表当前Servlet在web.xml中的配置信息。通过web.xml中加载我们自己写的MyDispatcherServlet和读取配置文件。

**5.3、初始化阶段**

我们知道了DispatcherServlet的initStrategies方法会初始化9大组件，但是本文将实现一些SpringMVC的最基本的组件而不是全部，按顺序包括：

·加载配置文件

·扫描用户配置包下面所有的类

·拿到扫描到的类，通过反射机制，实例化。并且放到ioc容器中(Map的键值对 beanName-bean) beanName默认是首字母小写

·初始化HandlerMapping，这里其实就是把url和method对应起来放在一个k-v的Map中,在运行阶段取出。

**5.4、运行阶段**

每一次请求将会调用doGet或doPost方法，所以统一运行阶段都放在doDispatch方法里处理，它会根据url请求去HandlerMapping中匹配到对应的Method，然后利用反射机制调用Controller中的url对应的方法，并得到结果返回。按顺序包括以下功能：

·异常的拦截。

·获取请求传入的参数并处理参数。

·通过初始化好的handlerMapping中拿出url对应的方法名，反射调用。