## MyBatis

### MyBatis原始使用（mybatis-1）

（1）准备：映射文件





（2）通过sqlsession使用

|  |
| --- |
| *//1.读取配置文件* InputStream in = Resources.getResourceAsStream(**"SqlMapConfig.xml"**); *//2.创建SqlSessionFactory工厂* SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(in); *//3.使用工厂生产SqlSession对象* SqlSession session = sqlSessionFactory.openSession(); *//4.执行Sql语句* User user = session.selectOne(**"test.findUserById"**, 10); *//5. 打印结果* System.out.println(user); *//6.释放资源* session.close(); in.close(); |

### 基于原始DAO（mybatis-2）

通过引入dao层将SqlSession 以及相应的SQL操作进行封装而已。

1. 准备：mapper文件如上

Dao层封装如下：

|  |
| --- |
| **private** SqlSessionFactory **sqlSessionFactory**; *// 通过构造方法注入* **public** UserDaoImpl(SqlSessionFactory sqlSessionFactory){  **this**.**sqlSessionFactory** = sqlSessionFactory; } **public** User findUserById(Integer id) {  *//sqlSession是线程不安全的，所以它的最佳使用范围在方法体内* SqlSession sqlSession = **sqlSessionFactory**.openSession();  User user = sqlSession.selectOne(**"test.findUserById"**,id);  sqlSession.close();  **return** user; } |

1. 使用

|  |
| --- |
| **private** SqlSessionFactory **factory**; **private** UserDao **userdao**; **private** InputStream **in**; *// 作用：在测试方法前执行这个方法* @Before **public void** setUp() **throws** Exception{  *//1.读取配置文件* **in** = Resources.*getResourceAsStream*(**"SqlMapConfig.xml"**);  *//2.创建SqlSessionFactory工厂* **factory** = **new** SqlSessionFactoryBuilder().build(**in**); }  **public void** testFindUserById() **throws** Exception{  *// 将初始化好的工厂注入到实现类中* UserDao userdao = **new** UserDaoImpl(**factory**);  User user = userdao.findUserById(34);  System.***out***.println(user);  *//User [id=34, username=小小, sex=1, birthday=Mon Jul 15 00:00:00 CST 2019, address=西安市]* } |

### 基于代理DAO（mybatis-3）

原始DAO需要定义DAOImpl类，并且在这个类中有大量的sqlsession重复代码，通过代理DAO可以解决此问题。

1. 准备：

A：mapper文件user.xml

B：User.xml文件中的mapper文件必须是全路径名：

|  |
| --- |
| **<mapper namespace="com.hust.lw.dao.UserDao">** |

C：去掉UserDaoImp.java文件

1. 使用

只需要有user.xml这个mapper文件，以及DAO接口文件，其中DAO接口文件中的接口与Mapper文件中的statement id是通过mapper文件中的namespace的全路径名关联的，然后通过sqlsession来使用。

|  |
| --- |
| **private** SqlSessionFactory **factory**; **private** UserDao **userdao**; **private** InputStream **in**; **private** SqlSession **session**;  *// 作用：在测试方法前执行这个方法* @Before **public void** setUp() **throws** Exception {  *//1.读取配置文件* **in** = Resources.*getResourceAsStream*(**"SqlMapConfig.xml"**);  *//2.创建SqlSessionFactory工厂* SqlSessionFactoryBuilder builder = **new** SqlSessionFactoryBuilder();  *//3.创建SqlSession工厂对象* SqlSessionFactory factory = builder.build(**in**);  *//4.使用工厂生产SqlSession对象* **session** = factory.openSession();  *//5.创建Dao接口的代理对象* **userdao** = **session**.getMapper(UserDao.**class**); }  @Test **public void** testFindUserById() {  *//5.使用代理对象执行方法* User user = **userdao**.findUserById(1);  System.***out***.println(user); } |

Mybatis的基本使用，以及来龙去脉通过这三个例子得到了一个整体的理解，随后就是一些复杂sql的mapper文件编写，以及在springboot中使用mybatis。

<https://www.cnblogs.com/benjieqiang/p/11183580.html>

## SpringMVC

### 1、环境搭建及controller与jsp互传数据（SpringMVC-1）

1. 新建SpringMVC工程



1. 部署Tomcat环境





1. 缺少Spring MVC相关类库引用





完成如上的3步后，则可以将这个空的SpringMVC工程跑起来了。

参考：

IDEA建立Spring MVC Hello World 详细入门教程

<https://www.cnblogs.com/wormday/p/8435617.html>

1. 创建controller类，找不到servlet

出现了问题： javax.servlet 包找不到

解决： 将本地 Tomcat 服务器的目录下【lib】文件夹下的 servlet-api.jar 包拷贝到工程【lib】文件夹下，添加依赖

这里面的“添加依赖”则出现了误会

参考：

cannot resolve symbol 'Servlet'

<https://blog.csdn.net/ycfn97/article/details/101114915>

1. 其他步骤，参考：

Spring MVC【入门】就这一篇！

<https://www.jianshu.com/p/91a2d0a1e45a>

说明：

A、 指定controller的方法，要么特定指定，要么泛指

|  |
| --- |
| *<bean id="simpleUrlHandlerMapping"  class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">  <property name="mappings">  <props>   <!-- /hello 路径的请求交给 id 为 helloController 的控制器处理-->    <prop key="/hello">helloController</prop>  </props>  </property> </bean> <bean id="helloController" class="com.hust.lw.controller.HelloController"></bean>* |

|  |
| --- |
| *<!-- 扫描ccom.hust.lw.controller下的组件 -->* <**context:component-scan base-package="com.hust.lw.controller"**/> |

如上只能选一个

B、在用模型传参时，模型中的set方法实现中，必须this.userName和this.password这样的格式

C、 jsp向后台controller传参一共有如下几种方法：

jnhs-SpringMVC jsp页面向controller传递参数的五种方式

<https://www.cnblogs.com/jnhs/p/9961589.html>

D、 Controller向前端jsp回传数据的方法：原生API完成、使用modelandview

、使用model、使用modelAttribute、map（也是一种model）

### 2、web工程转maven工程（SpringMVC-1-maven）

遇到了如下问题：

（1）pom.xml显示unknown

右键pom.xml点击maven->reimport

（2）调整目录结构

其中applicationContext.xml和\*-servlet.xml放在resources下

web.xml放在webapp/WEB-INF下

<https://blog.csdn.net/lch520baby/article/details/89374819>

（3）修改\*-servlet.xml路径

<https://blog.csdn.net/weixin_34194702/article/details/92248296>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:dispatcher-servlet.xml</param-value>

</init-param>

（4）jsp文件始终找不到，对比观察得知webapp文件夹上没有小蓝点

关于idea中新建web项目 webapp文件夹没有小蓝点 ，启动服务，访问不到解决方案

（5）获取各种路径

System.out.println(httpServletRequest.getContextPath());

System.out.println(httpServletRequest.getServletPath());

System.out.println(httpServletRequest.getRequestURI());

System.out.println(httpServletRequest.getRealPath("/"));

### 3、史上最全最强SpringMVC详细示例（SpringMVC-2）

<https://www.cnblogs.com/sunniest/p/4555801.html#commentform>

1. JSP到controller格式转换

由于前台传到controller里的值是String类型的，当往Model里Set这个值的时候，如果set的这个属性是个对象，Spring就会去找到对应的editor进行转换，然后再SET进去。

有两种：只在当前controller中生效的；在全局controller中生效的。

|  |
| --- |
| */\*\*  \* 使用InitBinder来处理Date类型的参数  \* the parameter was coverted in initBinder  \** ***@url:*** *http://localhost:8080/spring-mvc-action-tutorial/mvc/date?date=2016-05-17  \* http://localhost:8080/spring-mvc-action-tutorial/mvc/date?date=2016-05  \* http://localhost:8080/spring-mvc-action-tutorial/mvc/date?date=2016-05-17 15:00:00  \*   \** ***@param date*** *\** ***@return*** *\*/* @RequestMapping(**"/date"**) **public** String date(Date date) {  System.***out***.println(date);  **return "hello"**; }  */\*\*  \* At the time of initialization,* ***convert the type "String" to type "date"***  *\*   \** ***@param binder*** *\*/* @InitBinder **public void** initBinder(ServletRequestDataBinder binder) {  System.***out***.println(**"InitBinder of date : convert the type of String to type date."**);  binder.registerCustomEditor(Date.**class**, **new** CustomDateEditor(**new** SimpleDateFormat(**"yyyy-MM-dd"**), **true**)); } |

参考：

SpringMVC之自定义类型转换WebBindingInitializer@initBinder

<https://blog.csdn.net/zmx729618/article/details/54133830>

spring中@InitBinder和WebBindingInitializer的使用

<https://blog.csdn.net/yu870646595/article/details/51191188/>

1. 前端ajax调用到后台controller，然后返回数据到ajax

通过PrintWriter返回

|  |
| --- |
| */\*\*  \* 使用Ajax调用: pass the parameters to front-en using ajax  \* url: http://localhost:8080/spring-mvc-action-tutorial/mvc/page/ajax  \*   \** ***@param name*** *\** ***@param pw*** *\*/* @RequestMapping(value = **"/getPerson"**, method = RequestMethod.***POST***) **public void** getPerson(String name, PrintWriter pw) {  *// 也可以使用HttpServletResponse.getWriter()来获取PrintWriter* pw.write(**"hello, "** + name); } |

参考：

前后台数据交换，printwriter、jsonobject、jsonarray、ajax请求，数据交换

<https://www.hellojava.com/a/69745.html>

1. JSON在前后台之间传递

“客户端请求的数据是json字符串,springmvc不能直接解析，加@requestBody解析成对象，客户端请求的是key/value的拼接不需要加@requestBody，@responseBody返回json字符串，如果是ajax请求，我们想到得到json对象的话(为了方便根据key取value)”



对于request是json串，则必须要有@requestbody，如下图所示：



对于表单数据，则可以不需要@RequestBody：



Form-data、x-www-form-unlencoded、raw/json三者区别：







参考：

@RequestBody和@ResponseBody的用法以及Stringify()的作用

<https://blog.csdn.net/linlinlinfeng/article/details/82705772>

springmvc实现json交互 -requestBody和responseBody

<https://www.cnblogs.com/cnblog-long/p/6547380.html>

浅谈@RequestMapping @ResponseBody 和 @RequestBody 注解的用法与区别

<https://blog.csdn.net/ff906317011/article/details/78552426>

关于postman，参考如下两篇：

postman Params和Body的区别

<https://blog.csdn.net/Loren_SUN/article/details/86063374>

Postman使用详解

<https://www.cnblogs.com/xiaoxi-3-/p/7839278.html>

1. spring异常统一处理

Spring MVC异常统一处理的三种方式

<https://www.cnblogs.com/junzi2099/p/7840294.html>

1. SpringIOC和SpingMVC整合

主要就是一些配置文件的修改。



参考：（@RequestBody和@ModelAttribute两个同时使用时，以@ModelAttribute为主）

@RequestParam、@RequestBody和@ModelAttribute区别

<https://www.cnblogs.com/zeroingToOne/p/8992746.html>

|  |
| --- |
| **(1)spring 中bean的作用域**  Application scope----web服务器启动后变量或实例就存在于内存中，除非重启服务，否则不会初始化。  Session Scope ----一次回话存在于内存中，就是你开一次新的网页就初始化一次。有20钟有效期！20分钟内都是有效的变量或实例，只要在20分钟内，session的变量可以随便调用（与Request Scope 的区别是，Request Scope是在请求结束后就失效了）[结合controller的session形参对象]  Request Scope ----是完成请求后，变量或实例就失效了。不能再次调用  Page Scope ----变量或实例只在当前页面有效  在web.xml中加入  <session-config>  <session-timeout>100</session-timeout>  </session-config>  以分为单位  **（2）SpringMVC模型model**  SpringMVC在调用方法前会创建一个隐含的数据模型，作为模型数据的存储容器，成为”隐含模型”。也就是说在每一次的前后台请求的时候会随带这一个背包，不管你用没有，这个背包确实是存在的，用来盛放我们请求交互传递的值。  有三种模型：**Model、ModelMap和ModelAndView**,前两者是模型，最后一个是模型+视图。  Spring中Model详解  <https://blog.csdn.net/yongwa123/article/details/85017551>   1. **使用模型Model**   @ModelAttribute最主要的作用是将数据添加到模型对象中，用于视图页面展示时使用。  @ModelAttribute等价于 model.addAttribute("attributeName", abc);  @ModelAttribute注解的使用总结  <https://blog.csdn.net/leo3070/article/details/81046383> |

1. 其他重要参考

4.1 吐血整理！14 个 Spring MVC 顶级技巧！

<https://blog.csdn.net/bntX2jSQfEHy7/article/details/104980512>

1. 多种控制类
2. 基于类的控制接口和基于方法的控制接口
3. 将对象放入模型中

4.2 SpringMVC-方法四种类型返回值总结，你用过几种？

<https://www.cnblogs.com/lenve/p/10817619.html>

1. 前后端未分离时ModelAndView用的比较多，即返回数据也返回视图
2. Void也就是各种控制跳转，String，JSON

4.3 Spring MVC教程（易佰）

## SSM

### 1、详解idea搭建SSM框架（SSM-1）

(1)完整搭建

SQLyog连接虚拟机MySQL连不上解决

<https://www.cnblogs.com/invban/p/5824891.html>

<https://www.cnblogs.com/linjiqin/p/5270938.html>

右侧没有mybatis-generator选项

<https://blog.csdn.net/yytwiligt/article/details/81010360>

详解intellij idea搭建SSM框架(spring+maven+mybatis+mysql+junit)(上)

<https://www.cnblogs.com/toutou/p/ssm_spring.html>

（2）session理解



Session：将user数据保存在服务端，通过一个jsessionid将前端和后台匹配起来，分布式后台存在session同步的问题

Token：所有的信息在token中，由前端保存，后端仅签名和验签

Postman使用jsessionid：只要是一个合法的接口，都会返回sessionid，但是这个sessionid不是合法的（不对应用户），只有在先调用login接口后返回的sessionid才是合法的，后续带着这个sessionid去访问，后台才认为合法， 否则500内部错误

参考：

SpringMVC HttpSession 与 Model 区别

<https://blog.csdn.net/Hello_ChenLiYan/article/details/106411452>

postman测试带sessionid的接口

<https://blog.csdn.net/maryshine/article/details/81014887>

### 2、前后端分离和分页查询（SSM-2）

（1）前后端分离，接收的数据是Json时需要@RequestBody作用在方法参数上.用于将POST请求数据转化为实体类.返回数据时使用@ResponseBody;

当然对于全局的也可以使用@RespController注解.前后端不分离一般使用@RequestMapping，请求方式不用管。

|  |
| --- |
| @Controller  Public class shopManagerController{  @RequestMapping(value="/getshopmanagementinfo",method=RequestMethod.GET) @ResponseBody  private Map<String,Object> getShopManagementInfo(@RequestBody User user){  } |

|  |
| --- |
| @RestController  Public class shopManagerController{  @RequestMapping(value="/getshopmanagementinfo",method=RequestMethod.GET)  private Map<String,Object> getShopManagementInfo(@RequestBody User user){  } |

（2）A:如果只是使用@RestController注解Controller，则Controller中的方法无法返回jsp页面，配置的视图解析器InternalResourceViewResolver不起作用，返回的内容就是Return 里的内容。

例如：本来应该到success.jsp页面的，则其显示success.

B:如果需要返回到指定页面，则需要用 @Controller配合视图解析器InternalResourceViewResolver才行。

C:如果需要返回JSON，XML或自定义mediaType内容到页面，则需要在对应的方法上加上@ResponseBody注解

(3)

@RestController注解相当于@ResponseBody ＋ @Controller合在一起的作用。

@GetMapping是一个组合注解，是@RequestMapping(method = RequestMethod.GET)的缩写。

@PostMapping是一个组合注解，是@RequestMapping(method = RequestMethod.POST)的缩写。

（4）示例1

|  |
| --- |
| @RequestMapping(value="hello",method=RequestMethod.*POST*) @ResponseBody Map<String, Object> hello(@RequestBody User user){  Map<String, Object> hashmap = new HashMap<String, Object>();  hashmap.put("status","success");  hashmap.put("code",200);  hashmap.put("user",user);  return hashmap; } |



（5）示例2

|  |
| --- |
| @RequestMapping(value="hello2",method=RequestMethod.*POST*) @ResponseBody RestBean hello2(@RequestBody User user){  RestBean restBean = RestBean.*ok*("this is ok");  restBean.setObj((Object)user);  return restBean; } |



SSM前后端分离/不分离对比Demo

<https://www.cnblogs.com/liruilong/p/12244925.html>

SSM框架中的前后端分离【详细说明了前后端分离的意义及示例】

<https://www.cnblogs.com/Tu9oh0st/p/10147168.html>

Jackson和data注解

<http://www.luyixian.cn/news_show_6846.aspx>

<https://blog.csdn.net/zzzghj/article/details/88828984>

<https://www.jianshu.com/p/484c039930e2>

Mysql