

高级 WEB LAB1

——学习 SPRING MVC

侯天朗 13302010024

Spring 与 hibernate 介绍

1. Spring 是一个开源框架，是为了解决企业应用程序开发复杂性而创建的。框架的主要优势之一就是其分层架构，分层架构允许您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。

2. 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，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。

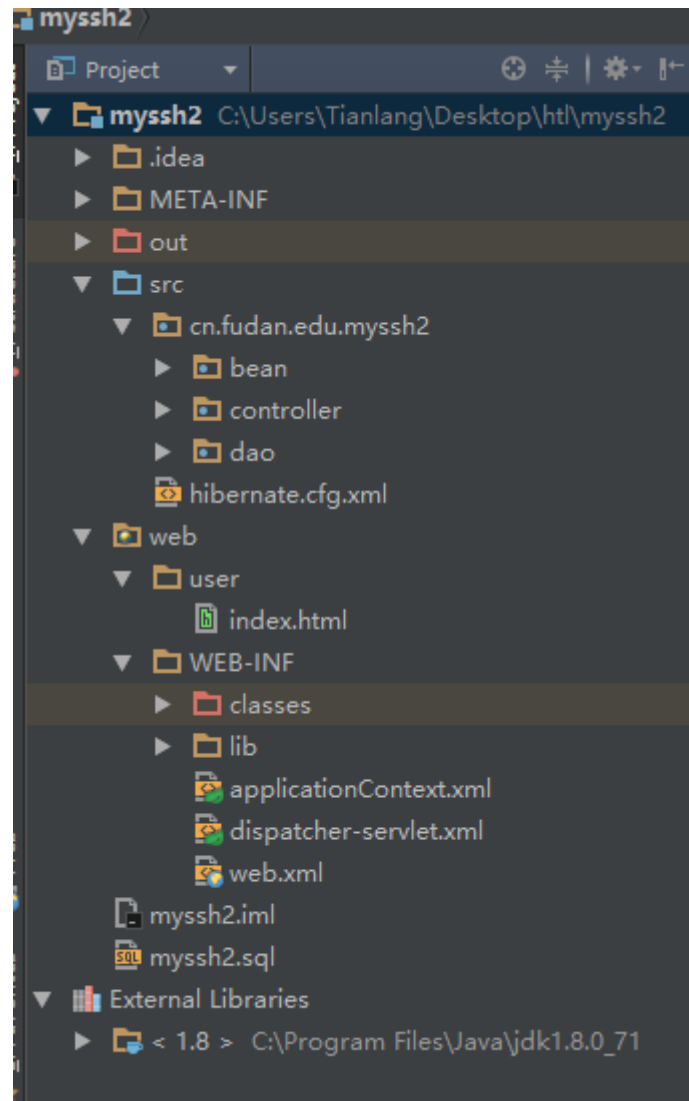
核心架构的具体流程步骤如下：

- 1、 首先用户发送请求——>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， HandlerMapping 将会把请求映射为 HandlerExecutionChain 对象（包含一个 Handler 处理器（页面控制器）对象、多个 HandlerInterceptor 拦截器）对象，通过这种策略模式，很容易添加新的映射策略；
- 3、 DispatcherServlet——>HandlerAdapter， HandlerAdapter 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——>处理器功能处理方法的调用， HandlerAdapter 将会根据适配的结果调用真正的处理器的功能处理方法，完成功能处理；并返回一个 ModelAndView 对象（包含模型数据、逻辑视图名）；
- 5、 ModelAndView 的逻辑视图名——> ViewResolver， ViewResolver 将把逻辑视图名解析为具体的 View，通过这种策略模式，很容易更换其他视图技术；
- 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，因此很容易支持其他视图技术；
- 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。

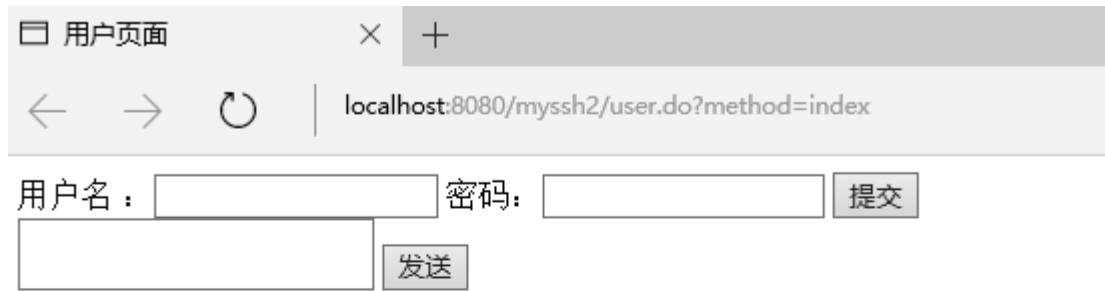
项目介绍

简单的用户登录系统

项目结构



项目演示



用户页面 × +

localhost:8080/myssh2/user.do?method=index

用户名: 密码:

具体实现

配置 spring mvc

```
xmlns:context="http://www.springframework.org/schema/context"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans http://www.springframework.org/schema/context http://www.springframework.org/schema/context http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc"

<!-- 指明 controller 所在包，并扫描其中的注解 -->
<context:component-scan base-package="cn.fudan.edu.myssh2.controller"/>

<!-- 静态资源(js、image等)的访问 -->
<mvc:default-servlet-handler/>

<!-- 开启注解 -->
<mvc:annotation-driven/>

<!-- ViewResolver 视图解析器 -->
<!-- 用于支持Servlet、JSP视图解析 -->
<bean id="jspViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/" />
    <property name="suffix" value=".html" />
</bean>
```

配置 hibernate

```

<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:mysql://127.0.0.1:3306/sshtest</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.username">root</property>
        <property name="connection.password">941106nt</property>

        <!-- 配置hibernate的基本信息 -->

        <!-- 配置数据库方言dialect -->
        <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
        <!-- 指定自动生成数据表的策略 -->
        <property name="hbm2ddl.auto">update</property>
        <mapping class="cn.fudan.edu.myssh2.bean.User"/>
        <mapping class="cn.fudan.edu.myssh2.bean.Message"/>

```

当用户访问网站时，拦截器

```

/**
 * Created by Sissors-CX on 2016-04-07.
 */
@Controller
@RequestMapping(value = "/")
public class DispatcherController {

    @RequestMapping(method = RequestMethod.GET)
    public String index() { return "redirect:user.do?method=index"; }
}

```

1. 在 web.xml 中配置 DispatcherServlet

DispatcherServlet 是 Spring MVC 的灵魂和心脏，它负责接收 HTTP 请求并协调 Spring MVC 的各个组件完成请求处理的工作。和任何 Servlet 一样，用户必须在 web.xml 中配置好 DispatcherServlet。如下：

```

web app | servlet | servlet class
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>
<listener>
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<servlet>
<servlet-name>dispatcher</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>dispatcher</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>
<filter>
<filter-name>encodingFilter</filter-name>
<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
</filter>

```

2. 编写处理请求的控制器

Spring MVC 通过一个@Controller 注解即可将一个 POJO 转化为处理器请求的控制器，通过@RequestMapping 为控制器指定处理哪些 URL 的请求。

```

@Controller
@RequestMapping(value = "/message.do")
public class MessageController {

    @RequestMapping(params = "method=send")
    @ResponseBody
    public String receive(@RequestBody String request) throws JSONException {
        JSONObject requestJson = new JSONObject(request);
        Message message = new Message();
        message.setSource(0);
        message.setContent(requestJson.get("content").toString());
        message.setTime(new Timestamp(System.currentTimeMillis()));
        MessageDao.add(message);
        JSONObject response = new JSONObject();
        response.put("status", "success");
        return response.toString();
    }
}

```

3. 编写视图对象

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>用户页面</title>
    <script src="//cdn.bootcss.com/jquery/2.2.1/jquery.min.js"></script>
  </head>
  <body>
    <form action="user.do">
      用户名 : <input type="text" name="name"/>
      密码: <input type="password" name="password"/>
      <input type="hidden" name="method" value="register"/>
      <button type="submit">提交</button>
    </form>
    <textarea id="message"></textarea>
    <button type="button" id="send">发送</button>
    <script>
      $("#send").click(function () {

```

SpringMVC 处理 index.html 过程描述

- (1) DispatcherServlet 接收到客户端的 user.html 的请求
- (2) DispatcherServlet 使用 DefaultAnnotationHandlerMapping 查找负责处理该请求的处理器为 user.html
- (3) DispatcherServlet 将请求分发给名为 user.html 的 UserController 处理器
- (4) 处理器完成业务处理后，返回 ModelAndView 对象，其中 View 的逻辑名为 menu，模型包含一个键为 user 的 User 对象
- (5) DispatcherServlet 调用 InternalResourceViewResolver 组件对 ModelAndView 中的逻辑视图名进行解析，得到真实的 /WEB-INF/customer/menu.jsp 视图对象
- (6) DispatcherServlet 使用 /WEB-INF/customer/menu.jsp 对模型中的 user 模型对象进行渲染
- (7) 返回响应页面给客户端

附源代码: <https://github.com/TianlangHou/myssh2.git>