# idea整合spring+spring mvc+mybatis+activiti

spring 配置文件：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx"  xsi:schemaLocation="http://www.springframework.org/schema/beans   http://www.springframework.org/schema/beans/spring-beans-4.0.xsd  http://www.springframework.org/schema/context   http://www.springframework.org/schema/context/spring-context-4.0.xsd  http://www.springframework.org/schema/aop   http://www.springframework.org/schema/aop/spring-aop-4.0.xsd  http://www.springframework.org/schema/tx   http://www.springframework.org/schema/tx/spring-tx-4.0.xsd ">    <!-- 1.扫描 service,dao -->  <context:component-scan base-package="com.activiti.service"/>  <!-- 2.引入属性文件，方便配置内容修改 -->  <context:property-placeholder location="classpath:jdbc.properties"/>    <!-- 3.数据库链接 -->  <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">  <property name="driverClass" value="${jdbc.driverClassName}"/>  <property name="jdbcUrl" value="${jdbc.url}"/>  <property name="user" value="${jdbc.username}"/>  <property name="password" value="${jdbc.password}"/>   <property name="maxPoolSize" value="${c3p0.pool.maxPoolSize}"/>  <property name="minPoolSize" value="${c3p0.pool.minPoolSize}" />  <property name="initialPoolSize" value="${c3p0.pool.initialPoolSize}"/>  <property name="acquireIncrement" value="${c3p0.pool.acquireIncrement}"/>  </bean>     <!-- 4.spring整合mybatis -->  <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  <property name="dataSource" ref="dataSource"/>  <property name="configLocation" value="classpath:mybatis/sqlMapConfig.xml"/>  <!-- 自动扫描映射文件 -->  <property name="mapperLocations" value="classpath:mapper/\*.xml" />  </bean>   <!--整合工作流-->  <import resource="classpath:activiti/activiti.cfg.xml"/>   <!-- 7.事务 -->   <bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  <property name="dataSource" ref="dataSource"/>  </bean>   <tx:advice id="txAdvice" transaction-manager="txManager">  <tx:attributes>  <tx:method name="insert\*" propagation="REQUIRED" />  <tx:method name="update\*" propagation="REQUIRED" />  <tx:method name="delete\*" propagation="REQUIRED" />  <tx:method name="find\*" read-only="true" />  <tx:method name="get\*" read-only="true" />  <tx:method name="view\*" read-only="true" />  </tx:attributes>  </tx:advice>   <aop:config>  <!-- 切点 管理所有Service的方法 -->  <aop:pointcut expression="execution(\* com.activiti.service.\*Service.\*(..))" id="transactionPointCut"/>  <!-- 增强，进行事务控制 Advisor -->  <aop:advisor advice-ref="txAdvice" pointcut-ref="transactionPointCut"/>  </aop:config> </beans> |

Spring-mvc配置文件：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:p="http://www.springframework.org/schema/p"  xsi:schemaLocation="http://www.springframework.org/schema/beans   http://www.springframework.org/schema/beans/spring-beans.xsd   http://www.springframework.org/schema/mvc   http://www.springframework.org/schema/mvc/spring-mvc.xsd   http://www.springframework.org/schema/context   http://www.springframework.org/schema/context/spring-context.xsd">   <!-- 配置自动扫描Controller -->  <context:component-scan  base-package="com.activiti.controller"  use-default-filters="false">  <context:include-filter type="annotation"  expression="org.springframework.stereotype.Controller" />  <context:include-filter type="annotation"  expression="org.springframework.web.bind.annotation.ControllerAdvice" />  </context:component-scan>   <!-- 注解驱动 -->  <mvc:annotation-driven>  <mvc:message-converters>  <!-- fastjson配置 -->  <bean  class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter">  <property name="supportedMediaTypes">  <list>  <value>text/html;charset=UTF-8</value>  <value>application/json;charset=UTF-8</value>  </list>  </property>  <property name="fastJsonConfig">  <bean class="com.alibaba.fastjson.support.config.FastJsonConfig">  <property name="dateFormat" value="yyyy-MM-dd HH:mm:ss" />  </bean>  </property>  </bean>  </mvc:message-converters>  </mvc:annotation-driven>   <!--&lt;!&ndash; 解决跨域 &ndash;&gt;-->  <!--<mvc:cors>-->  <!--<mvc:mapping path="/api/\*\*" -->  <!--allowed-origins="\*"-->  <!--allowed-methods="POST, GET, OPTIONS, DELETE, PUT"-->  <!--allowed-headers="Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With"-->  <!--/>-->  <!--</mvc:cors>-->   <!-- 当在web.xml 中 DispatcherServlet使用 <url-pattern>/</url-pattern> 映射时，能映射静态资源 -->  <mvc:default-servlet-handler/>   <!-- 静态资源映射 -->  <mvc:resources mapping="/static/\*\*" location="/WEB-INF/static"/>   <!-- 默认的视图解析器 在上边的解析错误时使用 (默认使用html)- -->  <bean id="defaultViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"  p:order="1">  <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>  <property name="contentType" value="text/html"/>  <property name="prefix" value="/WEB-INF/pages/"/>  <property name="suffix" value=".jsp"/>  </bean>    <!--文件上传视图解析器 -->  <bean id="multipartResolver"  class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  <!--允许上传的文件最大大小 单位是byte -->  <property name="maxUploadSize" value="60000000" />  </bean> </beans> |

Mybatis配置文件：sqlMapConfig.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"> <configuration>   <!-- 方言，所使用的数据库 -->  <properties>  <property name="dialect" value="mysql" />  </properties>   <settings>  <setting name="logImpl" value="LOG4J" />  </settings> </configuration> |

Activiti配置文件：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:tx="http://www.springframework.org/schema/tx"  xmlns:jee="http://www.springframework.org/schema/jee"  xmlns:aop="http://www.springframework.org/schema/aop"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.0.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd  http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd  http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-4.0.xsd  http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd">  <bean id="processEngineConfiguration"  class="org.activiti.spring.SpringProcessEngineConfiguration">  <!-- spring中配置的数据源，activiti的表和业务表在同一个数据库中 -->  <property name="dataSource" ref="dataSource"/>  <!-- spring中配置的事务，统一事务 -->  <property name="transactionManager" ref="txManager" />  <!-- 设置建表策略 true/create-drop/drop-create 删除重建，当表数据由于测试混乱，方便重建 -->  <property name="databaseSchemaUpdate" value="true" />  </bean>  <!-- 和spring集成，交给spring管理，由spring容器创建一个单例 -->  <bean id="processEngine" class="org.activiti.spring.ProcessEngineFactoryBean">  <property name="processEngineConfiguration" ref="processEngineConfiguration" />  </bean>  <!-- spring管理相关的服务 -->  <bean id="repositoryService" factory-bean="processEngine" factory-method="getRepositoryService" />  <bean id="runtimeService" factory-bean="processEngine" factory-method="getRuntimeService" />  <bean id="taskService" factory-bean="processEngine" factory-method="getTaskService" />  <bean id="formService" factory-bean="processEngine" factory-method="getFormService" />  <bean id="historyService" factory-bean="processEngine" factory-method="getHistoryService" /> </beans> |

jdbc.properties 配置文件：

|  |
| --- |
| jdbc.driverClassName=com.mysql.jdbc.Driver jdbc.url=jdbc:mysql://localhost:3306/activiti?characterEncoding=utf-8 jdbc.username=root jdbc.password=mysqladmin  c3p0.pool.maxPoolSize=20 c3p0.pool.minPoolSize=5 c3p0.pool.initialPoolSize=3 c3p0.pool.acquireIncrement=2 |

log4j.properties：

|  |
| --- |
| log4j.rootLogger=Info,stdout #Console log4j.appender.stdout=org.apache.log4j.ConsoleAppender log4j.appender.stdout.layout=org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern=%-5p - %m%n #LogFile log4j.appender.file=org.apache.log4j.FileAppender log4j.appender.file.File=../logs/activiti.log log4j.appender.file.layout=org.apache.log4j.PatternLayout log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n #Daily LogFile log4j.appender.dayfile=org.apache.log4j.DailyRollingFileAppender log4j.appender.dayfile.File=../logs/activiti.log log4j.appender.dayfile.DatePattern='.'yyyy-MM-dd'.log' log4j.appender.dayfile.Append=true log4j.appender.dayfile.layout=org.apache.log4j.PatternLayout log4j.appender.dayfile.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n # LOGIN\_LOG\_P db config ### log4j.appender.logindb=org.apache.log4j.jdbc.JDBCAppender log4j.appender.logindb.BufferSize=1 log4j.appender.logindb.driver=com.mysql.jdbc.Driver log4j.appender.logindb.URL=jdbc:mysql://localhost:3306/ktb?characterEncoding=utf-8 log4j.appender.logindb.user=root log4j.appender.logindb.password=mysqladmin  ## MySQL #log4j.appender.logindb.sql=INSERT INTO LOGIN\_LOG\_P(LOGIN\_LOG\_ID,LOGIN\_NAME,IP\_ADDRESS,LOGIN\_TIME) VALUES(uuid(),'%X{userId}|%X{userName}','%X{ip}','%d{yyyy-MM-dd HH:mm:ss}') log4j.appender.logindb.sql=INSERT INTO LOGIN\_LOG\_P(LOGIN\_LOG\_ID,LOGIN\_NAME,IP\_ADDRESS,LOGIN\_TIME) VALUES(uuid(),'','',NOW()) log4j.appender.logindb.layout=org.apache.log4j.PatternLayout log4j.logger.org.apache=INFO log4j.logger.com.activiti=DEBUG |

Web.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  version="3.1">  <!--配置spring-->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/beans.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 防止spring内存溢出监听器 -->  <listener>  <listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>  </listener>  <!-- 加载springMVC的核心分发器 DispatcherServlet -->  <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:my-spring-mvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  <async-supported>true</async-supported>  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>   <!-- 编码过滤器，解决中文乱码 -->  <filter>  <filter-name>SpringEncoding</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>SpringEncoding</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>   <welcome-file-list>  <welcome-file>/deployments</welcome-file>  </welcome-file-list> </web-app> |

## 实现的功能：

1. 实现获得部署流程的列表
2. 通过zip上传文件来部署流程
3. 读取对应的流程xml和png图片
4. 删除流程部署

### 代码：

### Common.jsp

|  |
| --- |
| <%@ **page** contentType="text/html;charset=UTF-8" language="java" isELIgnored="false" %> <%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <%@ **taglib** prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %> <**c:set** value="**${**pageContext.request.contextPath**}**" var="ctx"/> <**c:set** value="**${**ctx**}**/static"/> |

### Index.jsp:

|  |
| --- |
| <%@ **page** contentType="text/html;charset=UTF-8" language="java" %> <%@**include** file="base/common.jsp"%> <html> <head>  <title>流程部署</title>  <script src="**${**ctx**}**/static/js/jquery-3.2.1.min.js"></script> </head> <body>  <div style="width:900px;margin:50px auto;">  <h2>流程信息列表：</h2>  <table border="1" width="90%">  <tr>  <th>流程定义ID</th>  <th>流程部署Id</th>  <th>流程定义名称</th>  <th>流程定义KEY</th>  <th>版本号</th>  <th>XML资源文件</th>  <th>图片资源文件</th>  <th>操作</th>    </tr>  <**c:forEach** items="**${**dataList**}**" var="item">  <tr>  <td>**${**item.id**}**</td>  <td>**${**item.deploymentId**}**</td>  <td>**${**item.name**}**</td>  <td>**${**item.key**}**</td>  <td>**${**item.version**}**</td>  <td><a href="**${**ctx**}**/deployment?id=**${**item.deploymentId**}**&name=**${**item.resourceName**}**">**${**item.resourceName**}**</a></td>  <td><a href="**${**ctx**}**/deployment?id=**${**item.deploymentId**}**&name=**${**item.diagramResourceName**}**">**${**item.diagramResourceName**}**</a></td>  <td><a href="javascript:void(0)" onclick="deletes(**${**item.deploymentId**}**)">删除</a></td>  </tr>   </**c:forEach**>  </table>  <h2>上传压缩包部署流程</h2>  <form action="**${**ctx**}**/deployment" method="post" enctype="multipart/form-data">   <input type="file" name="file" id="file">  <input type="submit" value="上传">   </form>  </div> <script>   **function** deletes(id){  **var** index = confirm('是否删除');   **if**(index){  $.ajax({   url:'**${**ctx**}**/deployment/'+id,  type:'DELETE',  success: **function**(data){    data = JSON.parse(data);  **if**(data.success){  document.location.reload();  }**else**{  alert('删除失败');  }  }  });  }  }  </script> </body> </html> |

### Controller代码：

|  |
| --- |
| /*\*\*  \* Created by @author 王钟鑫  \*  \** ***@create*** *2018/1/20.  \*/* @Controller public class DeployManagementController {   private Logger log = Logger.*create*(DeployManagementController.class);   //流程部署实例  private RepositoryService repositoryService;   @Resource  public void setRepositoryService(RepositoryService repositoryService) {  this.repositoryService = repositoryService;  }   @RequestMapping(value = "/deployments",method = RequestMethod.*GET*)  public String index(Model model){   /\* repositoryService.createDeployment().addClasspathResource("diagram/HelloWorld.bpmn")  .addClasspathResource("diagram/HelloWorld.png").deploy();\*/   //获取到已经部署的流程  List<ProcessDefinition> dataList = repositoryService.createProcessDefinitionQuery().list();   model.addAttribute("dataList",dataList);   return "index";  }  /\*  \*  \* 通过上传的文件zip,bpmn,png文件来部署流程  \* @Param 前端form表单的file对象  \*  \* \*/   @RequestMapping(value = "/deployment",method = RequestMethod.*POST*)  public String deploy(@RequestParam("file")MultipartFile file){   if(file != null){   String originName = file.getOriginalFilename();  try {   InputStream inputStream = file.getInputStream();  //获取到扩展名  String type = FilenameUtils.*getExtension*(originName);   //创建部署对象  DeploymentBuilder deployment = repositoryService.createDeployment();   //如果是zip，bar类型需要使用Zip对象来进行部署  if(type.equals("zip") || type.equals("bar")){   ZipInputStream zip = new ZipInputStream(inputStream);  deployment.addZipInputStream(zip);   //其他文件直接部署  }else{   deployment.addInputStream(originName,inputStream);   }  //部署  deployment.deploy();   return "redirect:deployments";   } catch (IOException e) {   e.printStackTrace();   }   }  return "error";   }  /\*  \*  \* 获取到部署的xml或者是png图片  \* @Param deploymentId 部署Id  @Param resourceName 资源的名称  \*  \* \*/  @RequestMapping(value = "/deployment",method = RequestMethod.*GET*)  public void getXmlResource(@RequestParam("id") String deploymentId,@RequestParam("name") String resourceName, HttpServletResponse response){   //获取到查询对象  InputStream resourceAsStream = repositoryService.getResourceAsStream(deploymentId, resourceName);   byte[] b = new byte[1024];  int len = -1;  try {  while ((len = resourceAsStream.read(b,0,1024)) != -1){   response.getOutputStream().write(b,0,len);  }   } catch (IOException e) {  e.printStackTrace();  }    }   @ResponseBody  @RequestMapping(value = "/deployment/{id}",method = RequestMethod.*DELETE*)  public Object delete(@PathVariable("id")String deploymentId){   try {  repositoryService.deleteDeployment(deploymentId,true);  return R.*success*();  }catch (Exception e){   e.printStackTrace();   }   return R.*failure*();  } } |

界面展示：

