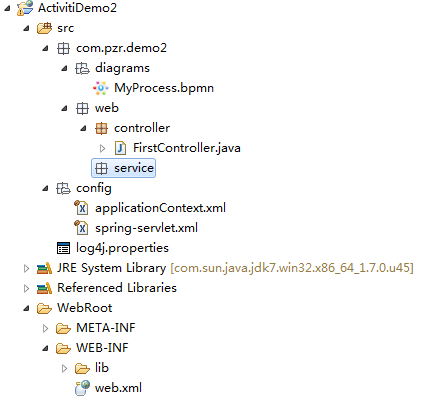
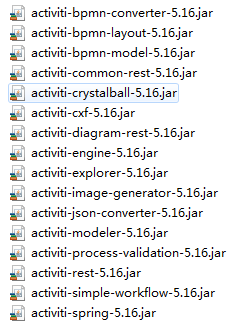
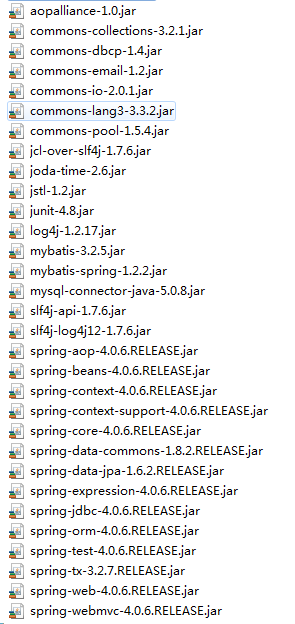
[Activiti与Spring集成](http://blog.csdn.net/flygoa/article/details/51882054)

1. 基础准备

## 目录结构



## 相关jar包

Activiti的相关jar包   
Activiti依赖的相关jar包   
[**spring**](http://lib.csdn.net/base/javaee)的相关jar包   
Spring依赖的相关jar包   
本示例相关jar包截图   
   


# 配置文件设置

## web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

id="WebApp\_ActivitiDemo2" version="2.5">

<display-name>ActivitiDemo2</display-name>

<!-- 监听： 在启动Web 容器时，自动装配Spring applicationContext.xml 的配置信息 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 加载spring主配置文件 -->

<context-param>

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

<param-value>classpath:config/applicationContext.xml</param-value>

</context-param>

<!-- 配置spring拦截器 -->

<servlet>

<servlet-name>springMvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

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

<!-- 配置请求路径 -->

<param-value>classpath:config/spring-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup><!-- load-on-startup必须放在最后 -->

</servlet>

<servlet-mapping>

<servlet-name>springMvc</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<!-- 转换为UTF-8编码 -->

<filter>

<filter-name>itxxzEncoding</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>

<init-param>

<param-name>forceEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>itxxzEncoding</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

</web-app>

## spring-servlet.xml

<?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"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd">

<!-- 启用spring mvc 注解 -->

<context:annotation-config />

<!-- 设置使用注解的类所在的jar包 controller扫描路径 -->

<context:component-scan base-package="com.pzr.demo2.web.controller"></context:component-scan>

<mvc:annotation-driven>

<!-- 消息转换器，配置字符集解决乱码问题 -->

<mvc:message-converters register-defaults="true">

<bean class="org.springframework.http.converter.StringHttpMessageConverter">

<property name="supportedMediaTypes" value="text/html;charset=UTF-8" />

</bean>

</mvc:message-converters>

</mvc:annotation-driven>

<!-- 使用jsp作为视图 -->

<bean id="viewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass"

value="org.springframework.web.servlet.view.JstlView" />

<property name="prefix" value="/WEB-INF/views/" />

<property name="suffix" value=".jsp"></property>

</bean>

</beans>

## applicationContext.xml

<?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-3.1.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.1.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.1.xsd

http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-3.1.xsd

http://www.springframework.org/schema/data/jpa http://www.springframework.org/schema/data/jpa/spring-jpa-1.0.xsd

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd">

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"

destroy-method="close">

<property name="driverClassName" value="com.mysql.jdbc.Driver">

</property>

<property name="url">

<value>jdbc:mysql://localhost:3306/actdemo1?useUnicode=true&amp;characterEncoding=utf-8</value>

</property>

<property name="username" value="root">

</property>

<property name="password" value="">

</property>

</bean>

<!-- 配置事务 -->

<bean id="txManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource"></property>

</bean>

<!--使用基于注解方式配置事务 -->

<tx:annotation-driven transaction-manager="txManager" />

<!-- activiti事务管理 -->

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

<!-- 加载activiti引擎 -->

<bean id="processEngine" class="org.activiti.spring.ProcessEngineFactoryBean">

<property name="processEngineConfiguration" ref="processEngineConfiguration" />

</bean>

<bean id="processEngineConfiguration" class="org.activiti.spring.SpringProcessEngineConfiguration">

<property name="dataSource" ref="dataSource" />

<property name="transactionManager" ref="transactionManager" />

<property name="databaseSchemaUpdate" value="true" />

<property name="jobExecutorActivate" value="false" />

</bean>

<!-- activiti的各种服务接口 -->

<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="historyService" factory-bean="processEngine"

factory-method="getHistoryService" />

<bean id="managementService" factory-bean="processEngine"

factory-method="getManagementService" />

</beans>

# 示例代码

编写一个controller来调用接口

FirstController.java

package com.pzr.demo2.web.controller;

import java.util.List;

import org.activiti.engine.ProcessEngine;

import org.activiti.engine.ProcessEngines;

import org.activiti.engine.RepositoryService;

import org.activiti.engine.RuntimeService;

import org.activiti.engine.TaskService;

import org.activiti.engine.task.Task;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

@Controller

@RequestMapping(value = "/first")

public class FirstController {

@RequestMapping(value = "/test1")

@ResponseBody

public String test(){

// 加载配置文件activiti.cfg.xml，创建引擎，如果出现null，可能原因

//1.加载路径不是根目录。

//2.依赖包不完全

// 获取配置文件后，引擎开始创建数据库。

ProcessEngine engine = ProcessEngines.getDefaultProcessEngine();

// 获取流程储存服务组件

RepositoryService rs = engine.getRepositoryService();

// 获取运行时服务组件

RuntimeService rse = engine.getRuntimeService();

// 获取流程中的任务TASK组件

TaskService ts = engine.getTaskService();

// 部署流程，只要是符合BPMN2规范的XML文件，理论上都可以被ACTIVITI部署

rs.createDeployment().addClasspathResource("com/pzr/demo2/diagrams/MyProcess.bpmn").deploy();

// 开启流程，myprocess是流程的ID

rse.startProcessInstanceByKey("myProcess");

// 查询历史表中的Task

List<Task> task = ts.createTaskQuery().list();

Task task1 = task.get(task.size()-1);

System.out.println("第一环节："+task1);

System.out.println("推动流程到下一环节："+task1);

ts.complete(task1.getId());

task1 = ts.createTaskQuery().executionId(task1.getExecutionId()).singleResult();

System.out.println("第二环节：" + task1);

return "测试成功";

}

@Autowired

RepositoryService repositoryService;

@Autowired

RuntimeService runtimeService;

@Autowired

TaskService taskService;

@RequestMapping(value = "/test2")

@ResponseBody

public String test2(){

StringBuffer sb = new StringBuffer();

// 部署流程，只要是符合BPMN2规范的XML文件，理论上都可以被ACTIVITI部署

repositoryService.createDeployment().addClasspathResource("com/pzr/demo2/diagrams/MyProcess.bpmn").deploy();

// 开启流程，myprocess是流程的ID

runtimeService.startProcessInstanceByKey("myProcess");

// 查询历史表中的Task

List<Task> task = taskService.createTaskQuery().list();

Task task1 = task.get(task.size()-1);

sb.append("第一环节："+task1 +"<br/>");

sb.append("推动流程到下一环节："+task1+"<br/>");

taskService.complete(task1.getId());

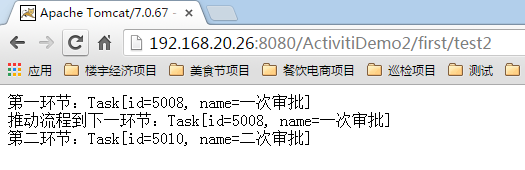
task1 = taskService.createTaskQuery().executionId(task1.getExecutionId()).singleResult();

sb.append("第二环节：" + task1+"<br/>");

return sb.toString();

}

}

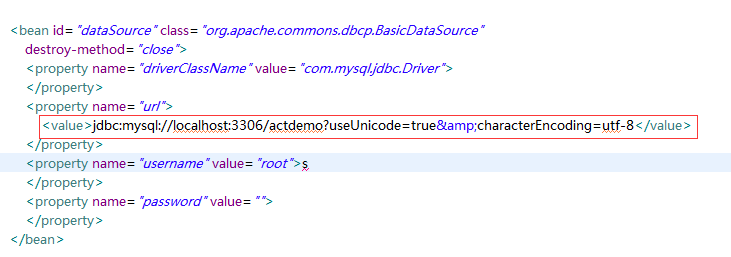
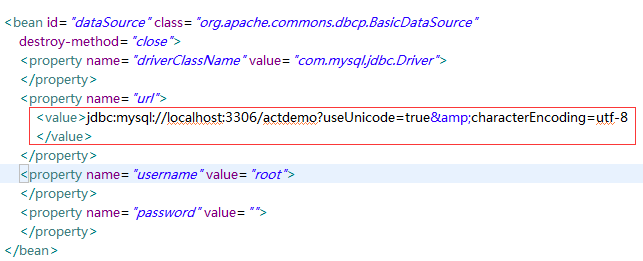
其中test1()和test2()是作为对比的，test2()中接口全是用注入的方式使用的   
把项目部署到tomcat中   
启动项目，会自动建表   
在浏览器运行<http://localhost:8080/ActivitiDemo2/first/test2>   
结果如下图：   


# 示例下载

[点击下载](http://pan.baidu.com/s/1o8D3H5S)

# 出现问题

Cannot create PoolableConnectionFactory (Unsupported character encoding ‘utf-8’.)

出现原因，将xml格式化化后，导致[**数据库**](http://lib.csdn.net/base/mysql)配置部分出现换行   
正常样子：   
   
格式化后（错误）样子：   
   
还原成正常样子即可解决问题