

Activiti整合Spring

一、Activiti与Spring整合开发

1.1 Activiti与Spring整合的配置

1)、在pom.xml文件引入坐标

如下

```
1  <properties>
2      <slf4j.version>1.6.6</slf4j.version>
3      <log4j.version>1.2.12</log4j.version>
4  </properties>
5  <dependencies>
6      <dependency>
7          <groupId>org.activiti</groupId>
8          <artifactId>activiti-engine</artifactId>
9          <version>7.0.0.Beta1</version>
10     </dependency>
11     <dependency>
12         <groupId>org.activiti</groupId>
13         <artifactId>activiti-spring</artifactId>
14         <version>7.0.0.Beta1</version>
15     </dependency>
16     <dependency>
17         <groupId>org.activiti</groupId>
18         <artifactId>activiti-bpmn-model</artifactId>
19         <version>7.0.0.Beta1</version>
20     </dependency>
21     <dependency>
22         <groupId>org.activiti</groupId>
23         <artifactId>activiti-bpmn-converter</artifactId>
24         <version>7.0.0.Beta1</version>
25     </dependency>
26     <dependency>
27         <groupId>org.activiti</groupId>
28         <artifactId>activiti-json-converter</artifactId>
29         <version>7.0.0.Beta1</version>
30     </dependency>
31     <dependency>
32         <groupId>org.activiti</groupId>
33         <artifactId>activiti-bpmn-layout</artifactId>
34         <version>7.0.0.Beta1</version>
35     </dependency>
36     <dependency>
37         <groupId>org.activiti.cloud</groupId>
```

```
38     <artifactId>activiti-cloud-services-api</artifactId>
39     <version>7.0.0.Beta1</version>
40 </dependency>
41 <dependency>
42     <groupId>aspectj</groupId>
43     <artifactId>aspectjweaver</artifactId>
44     <version>1.5.4</version>
45 </dependency>
46 <dependency>
47     <groupId>mysql</groupId>
48     <artifactId>mysql-connector-java</artifactId>
49     <version>5.1.40</version>
50 </dependency>
51 <dependency>
52     <groupId>junit</groupId>
53     <artifactId>junit</artifactId>
54     <version>4.12</version>
55 </dependency>
56 <dependency>
57     <groupId>org.springframework</groupId>
58     <artifactId>spring-test</artifactId>
59     <version>5.0.7.RELEASE</version>
60 </dependency>
61 <!-- log start -->
62 <dependency>
63     <groupId>log4j</groupId>
64     <artifactId>log4j</artifactId>
65     <version>${log4j.version}</version>
66 </dependency>
67 <dependency>
68     <groupId>org.slf4j</groupId>
69     <artifactId>slf4j-api</artifactId>
70     <version>${slf4j.version}</version>
71 </dependency>
72 <dependency>
73     <groupId>org.slf4j</groupId>
74     <artifactId>slf4j-log4j12</artifactId>
75     <version>${slf4j.version}</version>
76 </dependency>
77 <dependency>
78     <groupId>org.slf4j</groupId>
79     <artifactId>slf4j-nop</artifactId>
80     <version>${slf4j.version}</version>
81 </dependency>
82 <!-- log end -->
83 <dependency>
84     <groupId>org.mybatis</groupId>
85     <artifactId>mybatis</artifactId>
86     <version>3.4.5</version>
87 </dependency>
88 <dependency>
89     <groupId>commons-dbcp</groupId>
90     <artifactId>commons-dbcp</artifactId>
```

```

91         <version>1.4</version>
92     </dependency>
93 </dependencies>
94 <repositories>
95     <repository>
96         <id>alfresco</id>
97         <name>Activiti Releases</name>
98         <url>https://artifacts.alfresco.com/nexus/content/repositories/activiti-
releases/</url>
99         <releases>
100             <enabled>true</enabled>
101         </releases>
102     </repository>
103 </repositories>

```

在Activiti中核心类的是ProcessEngine流程引擎，与Spring整合就是让Spring来管理ProcessEngine

通过org.activiti.spring.SpringProcessEngineConfiguration 与Spring整合方式来创建ProcessEngine对象。

创建spring与activiti的整合配置文件：activiti-spring.xml（名称不固定）

2)、创建activiti-spring.xml

```

1  <beans xmlns="http://www.springframework.org/schema/beans"
2      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xmlns:tx="http://www.springframework.org/schema/tx"
4      xmlns:aop="http://www.springframework.org/schema/aop"
5      xsi:schemaLocation="http://www.springframework.org/schema/beans
6          http://www.springframework.org/schema/beans/spring-beans.xsd
7          http://www.springframework.org/schema/tx
8          http://www.springframework.org/schema/tx/spring-tx.xsd
9          http://www.springframework.org/schema/aop
10         http://www.springframework.org/schema/aop/spring-aop.xsd">
11      <!-- 数据源 -->
12      <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
13          <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
14          <property name="url" value="jdbc:mysql://localhost:3306/activiti"/>
15          <property name="username" value="root"/>
16          <property name="password" value="123456"/>
17          <property name="maxActive" value="3"/>
18          <property name="maxIdle" value="1"/>
19      </bean>
20      <!-- 工作流引擎配置bean -->
21      <bean id="processEngineConfiguration"
22          class="org.activiti.spring.SpringProcessEngineConfiguration">
23          <!-- 数据源 -->
24          <property name="dataSource" ref="dataSource"/>
25          <!-- 使用spring事务管理器 -->
26          <property name="transactionManager" ref="transactionManager"/>
27          <!-- 数据库策略 -->
28          <property name="databaseSchemaUpdate" value="drop-create"/>
29      </bean>
30      <!-- 流程引擎 -->
31      <bean id="processEngine" class="org.activiti.spring.ProcessEngineFactoryBean">

```

```

31     <property name="processEngineConfiguration" ref="processEngineConfiguration"/>
32 </bean>
33 <!-- 资源服务service -->
34 <bean id="repositoryService" factory-bean="processEngine" factory-
method="getRepositoryService"/>
35 <!-- 流程运行service -->
36 <bean id="runtimeService" factory-bean="processEngine" factory-
method="getRuntimeService"/>
37 <!-- 任务管理服务 -->
38 <bean id="taskService" factory-bean="processEngine" factory-method="getTaskService"/>
39 <!-- 历史管理服务 -->
40 <bean id="historyService" factory-bean="processEngine" factory-
method="getHistoryService"/>
41 <!-- 事务管理器 -->
42 <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
43     <property name="dataSource" ref="dataSource"/>
44 </bean>
45 <!-- 通知 -->
46 <tx:advice id="txAdvice" transaction-manager="transactionManager">
47     <tx:attributes>
48         <!-- 传播行为 -->
49         <tx:method name="save*" propagation="REQUIRED"/>
50         <tx:method name="insert*" propagation="REQUIRED"/>
51         <tx:method name="delete*" propagation="REQUIRED"/>
52         <tx:method name="update*" propagation="REQUIRED"/>
53         <tx:method name="find*" propagation="SUPPORTS" read-only="true"/>
54         <tx:method name="get*" propagation="SUPPORTS" read-only="true"/>
55     </tx:attributes>
56 </tx:advice>
57 <!-- 切面，根据具体项目修改切点配置 -->
58 <aop:config proxy-target-class="true">
59     <aop:advisor advice-ref="txAdvice"
60         pointcut="execution(*com.itheima.service.impl..(..))"/>
61 </aop:config-->
62 </beans>

```

databaseSchemaUpdate的取值内容：

flase：默认值。activiti在启动时，会对比数据库表中保存的版本，如果没有表或者版本不匹配，将抛出异常。（生产环境常用）
true：activiti会对数据库中所有表进行更新操作。如果表不存在，则自动创建。（开发时常用）
create_drop：在activiti启动时创建表，在关闭时删除表（必须手动关闭引擎，才能删除表）。（单元测试常用）
drop-create：在activiti启动时删除原来的旧表，然后在创建新表（不需要手动关闭引擎）。

1.2 测试Activiti与Spring整合

1)、测试代码

```
1  /**
```

```

2      测试activiti与spring整合是否成功
3      **/
4      @RunWith(SpringJUnit4ClassRunner.class)
5      @ContextConfiguration(locations = "classpath:activiti-spring.xml")
6      public class ActivitiTest {
7          @Autowired
8          private RepositoryService repositoryService;
9
10         @Test
11         public void test01(){
12             System.out.println("部署对象:"+repositoryService);
13         }
14     }

```

2)、执行流程分析

下面我们一起来分析Activiti与Spring整合加载的过程。

- 1、加载activiti-spring.xml配置文件
- 2、加载SpringProcessEngineConfiguration对象，这个对象它需要依赖注入dataSource对象和transactionManager对象。
- 3、加载ProcessEngineFactoryBean工厂来创建ProcessEngine对象，而ProcessEngineFactoryBean工厂又需要依赖注入processEngineConfiguration对象。
- 4、processEngine对象来负责创建我们的Service对象，从而简化Activiti的开发过程。

二、Activiti7与SpringBoot整合开发

Activiti7发布正式版之后，它与SpringBoot2.x已经完全支持整合开发。

2.1 SpringBoot整合Activiti7的配置

为了能够实现SpringBoot与Activiti7整合开发，首先我们要引入相关的依赖支持。

在工程的pom.xml文件中引入相关的依赖，其中activiti的依赖是：activiti-spring-boot-starter。

具体依赖如下所示：

```

1
2      <parent>
3          <groupId>org.springframework.boot</groupId>
4          <artifactId>spring-boot-starter-parent</artifactId>
5          <version>2.1.0.RELEASE</version>
6      </parent>
7      <properties>
8          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
9          <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
10         <java.version>1.8</java.version>

```

```

11 </properties>
12 <dependencies>
13     <dependency>
14         <groupId>org.springframework.boot</groupId>
15         <artifactId>spring-boot-starter-web</artifactId>
16     </dependency>
17     <dependency>
18         <groupId>org.springframework.boot</groupId>
19         <artifactId>spring-boot-starter-jdbc</artifactId>
20     </dependency>
21     <dependency>
22         <groupId>org.springframework.boot</groupId>
23         <artifactId>spring-boot-starter-test</artifactId>
24     </dependency>
25     <dependency>
26         <groupId>org.activiti</groupId>
27         <artifactId>activiti-spring-boot-starter</artifactId>
28         <version>7.0.0.Beta2</version>
29     </dependency>
30     <dependency>
31         <groupId>mysql</groupId>
32         <artifactId>mysql-connector-java</artifactId>
33         <version>5.1.29</version>
34     </dependency>
35     <dependency>
36         <groupId>org.projectlombok</groupId>
37         <artifactId>lombok</artifactId>
38     </dependency>
39 </dependencies>
40 <build>
41     <plugins>
42         <plugin>
43             <groupId>org.springframework.boot</groupId>
44             <artifactId>spring-boot-maven-plugin</artifactId>
45         </plugin>
46     </plugins>
47 </build>

```

通过该pom.xml文件所导入的坐标，我们就可以实现activiti7与Springboot整合。

2.2 SpringBoot的application.yml文件配置

为了能够实现Activiti7生成的表放到Mysql数据库中，需要在配置文件application.yml中添加相关的配置

注意：activiti7默认没有开启数据库历史记录，需要手动配置开启

```

1  spring:
2    datasource:
3      url: jdbc:mysql:///activiti?useUnicode=true&characterEncoding=utf8&serverTimezone=GMT
4      username: root
5      password: 123456
6      driver-class-name: com.mysql.jdbc.Driver

```

```

7      activiti:
8          #1.false: 默认值。activiti在启动时，对比数据库表中保存的版本，如果没有表或者版本不匹配，将抛出异常
9          #2.true:  activiti会对数据库中所有表进行更新操作。如果表不存在，则自动创建
10         #3.create_drop: 在activiti启动时创建表，在关闭时删除表（必须手动关闭引擎，才能删除表）
11         #4.drop-create: 在activiti启动时删除原来的旧表，然后在创建新表（不需要手动关闭引擎）
12         database-schema-update: true
13         #检测历史表是否存在 activiti7默认没有开启数据库历史记录 启动数据库历史记录
14         db-history-used: true
15         #记录历史等级 可配置的历史级别有none, activity, audit, full
16         #none: 不保存任何的历史数据，因此，在流程执行过程中，这是最高效的。
17         #activity: 级别高于none，保存流程实例与流程行为，其他数据不保存。
18         #audit: 除activity级别会保存的数据外，还会保存全部的流程任务及其属性。audit为history的默认值。
19         #full: 保存历史数据的最高级别，除了会保存audit级别的数据外，还会保存其他全部流程相关的细节数据，包括一些
            流程参数等。
20         history-level: full
21         #校验流程文件，默认校验resources下的processes文件夹里的流程文件
22         check-process-definitions: false

```

2.3 编写启动类

```

1  package com.itheima;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  public class ActApplication {
8      public static void main(String[] args) {
9          SpringApplication.run(ActApplication.class,args);
10     }
11 }
12

```

2.4 添加SpringSecurity安全框架整合配置

因为Activiti7与SpringBoot整合后，默认情况下，集成了SpringSecurity安全框架，这样我们就要去准备SpringSecurity整合进来的相关用户权限配置信息。

SpringBoot的依赖包已经将SpringSecurity的依赖包也添加进项目中。

2.4.1 添加SecurityUtil类

为了能够快速实现SpringSecurity安全框架的配置，所添加的一个组件。

```

1  package com.itheima.utils;
2
3  import org.slf4j.Logger;
4  import org.slf4j.LoggerFactory;
5  import org.springframework.beans.factory.annotation.Autowired;

```



```

58         public void setAuthenticated(boolean isAuthenticated) throws
    IllegalArgumentException { }
59         @Override
60         public String getName() {
61             return user.getUsername();
62         }
63     }));
64     org.activiti.engine.impl.identity.Authentication.setAuthenticatedUserId(username);
65 }
66 }

```

这个类可以从我们下载的Activiti7官方提供的Example中找到。

2.4.2 添加DemoApplicationConfig类

在Activiti7官方下载的Example中找到DemoApplicationConfig类，它的作用是为了实现SpringSecurity框架的用户权限的配置，这样我们就可以在系统中使用用户权限信息。

本次项目中基本是在文件中定义出来的用户信息，当然也可以是数据库中查询的用户权限信息。

后面处理流程时用到的任务负责人，需要添加在这里

```

1  package com.itheima.config;
2
3  import org.slf4j.Logger;
4  import org.slf4j.LoggerFactory;
5  import org.springframework.context.annotation.Bean;
6  import org.springframework.context.annotation.Configuration;
7  import org.springframework.security.core.authority.SimpleGrantedAuthority;
8  import org.springframework.security.core.userdetails.User;
9  import org.springframework.security.core.userdetails.UserDetailsService;
10 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
11 import org.springframework.security.crypto.password.PasswordEncoder;
12 import org.springframework.security.provisioning.InMemoryUserDetailsManager;
13
14 import java.util.Arrays;
15 import java.util.List;
16 import java.util.stream.Collectors;
17
18 @Configuration
19 public class DemoApplicationConfiguration {
20     private Logger logger = LoggerFactory.getLogger(DemoApplicationConfiguration.class);
21     @Bean
22     public UserDetailsService myUserDetailsService() {
23         InMemoryUserDetailsManager inMemoryUserDetailsManager = new
InMemoryUserDetailsManager();
24         //这里添加用户，后面处理流程时用到的任务负责人，需要添加在这里
25         String[][] usersGroupsAndRoles = {
26             {"jack", "password", "ROLE_ACTIVITI_USER", "GROUP_activitiTeam"},
27             {"rose", "password", "ROLE_ACTIVITI_USER", "GROUP_activitiTeam"},
28             {"tom", "password", "ROLE_ACTIVITI_USER", "GROUP_activitiTeam"},
29             {"other", "password", "ROLE_ACTIVITI_USER", "GROUP_otherTeam"},
30             {"system", "password", "ROLE_ACTIVITI_USER"},
31             {"admin", "password", "ROLE_ACTIVITI_ADMIN"},

```

```

32         };
33
34         for (String[] user : usersGroupsAndRoles) {
35             List<String> authoritiesStrings = Arrays.asList(Arrays.copyOfRange(user, 2,
user.length));
36             logger.info("> Registering new user: " + user[0] + " with the following
Authorities[" + authoritiesStrings + "]");
37             inMemoryUserDetailsManager.createUser(new User(user[0],
passwordEncoder().encode(user[1]),
38                 authoritiesStrings.stream().map(s -> new
SimpleGrantedAuthority(s)).collect(Collectors.toList())));
39         }
40
41         return inMemoryUserDetailsManager;
42     }
43     @Bean
44     public PasswordEncoder passwordEncoder() {
45         return new BCryptPasswordEncoder();
46     }
47 }

```

2.5 创建Bpmn文件

Activiti7可以自动部署流程，前提是在resources目录下，创建一个新的目录processes，用来放置bpmn文件。

创建一个简单的Bpmn流程文件，并设置任务的用户组Candidate Groups。

Candidate Groups中的内容与上面DemoApplicationConfiguration类中出现的用户组名称保持一致，可以填写：activitiTeam 或者 otherTeam。

这样填写的好处：当不确定到底由谁来负责当前任务的时候，只要是Groups内的用户都可以拾取这个任务

property	value
Id	_4
Name	UserTask
Documentation	
Asynchronous	<input type="checkbox"/>
Exclusive	<input checked="" type="checkbox"/>
Multi Instance	
Assignee	
Candidate Users	
Candidate Groups	activitiTeam
Due Date	
Form Key	
Priority	
Task Listeners	
Execution Listeners	
Form	

2.6 使用JUnit方式测试

```

1  package com.itheima.test;
2
3  import com.itheima.utils.SecurityUtil;
4  import org.activiti.api.process.model.ProcessInstance;
5  import org.activiti.api.process.model.builders.ProcessPayloadBuilder;
6  import org.activiti.api.process.runtime.ProcessRuntime;
7  import org.activiti.api.runtime.shared.query.Page;
8  import org.activiti.api.runtime.shared.query.Pageable;
9  import org.activiti.api.task.model.Task;
10 import org.activiti.api.task.model.builders.TaskPayloadBuilder;
11 import org.activiti.api.task.runtime.TaskRuntime;
12 import org.activiti.engine.repository.ProcessDefinition;
13 import org.junit.Test;
14 import org.junit.runner.RunWith;
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.boot.test.context.SpringBootTest;
17 import org.springframework.test.context.junit4.SpringRunner;
18
19 @RunWith(SpringRunner.class)
20 @SpringBootTest
21 public class Activiti7DemoApplicationTests {

```

```

22     @Autowired
23     private ProcessRuntime processRuntime;
24     @Autowired
25     private TaskRuntime taskRuntime;
26     @Autowired
27     private SecurityUtil securityUtil;
28
29     @Test
30     public void testActBoot(){
31         System.out.println(taskRuntime);
32     }
33
34     /**
35      * 查看流程定义
36      */
37     @Test
38     public void contextLoads() {
39         securityUtil.logInAs("system");
40         Page<org.activiti.api.process.model.ProcessDefinition> processDefinitionPage =
41             processRuntime.processDefinitions(Pageable.of(0, 10));
42         System.out.println("可用的流程定义数量: " + processDefinitionPage.getTotalItems());
43         for (org.activiti.api.process.model.ProcessDefinition pd :
processDefinitionPage.getContent()) {
44             System.out.println("流程定义: " + pd);
45         }
46     }
47
48
49     /**
50      * 启动流程实例
51      */
52     @Test
53     public void testStartProcess() {
54         securityUtil.logInAs("system");
55         ProcessInstance pi = processRuntime.start(ProcessPayloadBuilder.
56             start().
57             withProcessDefinitionKey("myProcess").
58             build());
59         System.out.println("流程实例ID: " + pi.getId());
60     }
61
62
63     /**
64      **查询任务，并完成自己的任务
65      **/
66     @Test
67     public void testTask() {
68         securityUtil.logInAs("jack");
69         Page<Task> taskPage=taskRuntime.tasks(Pageable.of(0,10));
70         if (taskPage.getTotalItems()>0){
71             for (Task task:taskPage.getContent()){
72                 taskRuntime.claim(TaskPayloadBuilder.
73                     claim().

```

```
74         withTaskId(task.getId()).build());
75         System.out.println("任务: "+task);
76         taskRuntime.complete(TaskPayloadBuilder.
77             complete().
78             withTaskId(task.getId()).build());
79     }
80 }
81 Page<Task> taskPage2=taskRuntime.tasks(Pageable.of*(0,10));
82 if (taskPage2.getTotalItems()>0){
83     System.out.println("任务: "+taskPage2.getContent());
84 }
85 }
86 }
```