Spring MVC

# 产品配置文件

## Web.xml

**//web.xml中通过contextConfigLocation定义spring的配置文件位置，类路径下spring/mt.oa.app.xml文件**

**//classpath：只会到class路径中查找文件;**

**//classpath\*：不仅包含class路径，还包括jar文件中(class路径)进行查找.**

<context-param>

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

<param-value>

classpath:spring/mt.oa.app.xml

</param-value>

</context-param>

**//Spring配置**

<listener>

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

</listener>

**//在web.xml中通过设置spring.profiles.default的值，定义使用的profile，在单元测试中用@ActiveProfiles(‘dev’)设置相应的dataSource**

<context-param>

<param-name>spring.profiles.default</param-name>

<param-value>${spring.profiles.active}</param-value>

</context-param>

**//Spring MVC 配置**

<servlet>

<servlet-name>springServlet</servlet-name>

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

**// contextConfigLocation指定Spring MVC 加载配置文件路径，默认加载/WEB-INF/[servlet-name]-servlet.xmlSpring，此处MVC框架将加载“spring/mt.oa.mvc.xml”来进行初始化上下文**

<init-param>

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

<param-value>classpath:spring/mt.oa.mvc.xml</param-value>

</init-param>

// **load-on-startup：表示启动容器时初始化该Servlet的顺序；**

<load-on-startup>1</load-on-startup>

</servlet>

// **url-pattern：表示哪些请求交给Spring Web MVC处理， “/” 为所有请求。也可以如“\*.html”表示拦截所有以html为扩展名的请求。**

<servlet-mapping>

<servlet-name>springServlet</servlet-name>

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

</servlet-mapping>

## mt.oa.app.xml主配置文件

<?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:jdbc="http://www.springframework.org/schema/jdbc" xmlns:jee="http://www.springframework.org/schema/jee"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd

http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd

http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.2.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd"

default-lazy-init="true">

<description>Spring公共配置 </description>

**<!-- base-package属性指定了需要扫描的类包，类包及其递归子包中所有的类都会被处理；exclude-filter：排除扫描Controller和ControllerAdvice，在mt.oa.mvc.xml中只扫描它两。-->**

<context:component-scan base-package="com.meituan.oa">

<context:exclude-filter type="annotation"

expression="org.springframework.stereotype.Controller" />

<context:exclude-filter type="annotation"

expression="org.springframework.web.bind.annotation.ControllerAdvice" />

</context:component-scan>

**<!--扫瞄所有的@Service和@Component -->**

<context:component-scan base-package="com.meituan.oa">

<context:include-filter type="annotation"

expression="org.springframework.stereotype.Service" />

</context:component-scan>

<context:component-scan base-package="com.meituan.oa">

<context:include-filter type="annotation"

expression="org.springframework.stereotype.Component" />

</context:component-scan>

|  |  |
| --- | --- |
| 事务管理实现 | 使用时机 |
| org.springframework.jdbc.datasource.  DataSourceTransactionManager | 在单一的JDBC DataSource中管理事务 |
| org.springframework.orm.hibernate3.  HibernateTransactionManager | 当持久化机制是Hibernate时，用它来管理职务 |
| org.springframework.orm.  jpa.JpaTransactionManager | 当JPA用作持久化时，用它来管理职务 |
| org.springframework.transaction.  jta.JtaTransactionManager | 使用一个JTA实现来管理事务。在一个事务跨越多个资源时必须使用 |

**//配置事务管理**

<bean id="transactionManager"

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

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

</bean>

**//配置邮件服务，MailBizServiceImpl中注入JavaMailSender，调用其send(MimeMessagePreparator mmp)方法**

<bean id="mail.send" class="org.springframework.mail.javamail.JavaMailSenderImpl">

<!-- SMTP settings -->

<property name="host" value="${smtp\_host}" />

<property name="port" value="${smtp\_port}" />

<property name="username" value="${mail\_user\_name}" />

<property name="password" value="${mail\_pwd}" />

<property name="javaMailProperties">

<!-- additional properties specific to JavaMail -->

<props>

<prop key="mail.transport.protocol">smtp</prop>

<prop key="mail.smtp.auth">true</prop>

<prop key="mail.smtp.starttls.enable">true</prop>

</props>

</property>

</bean>

<!-- ？？？？-->

<bean id="ajaxAuthFilter" class="com.meituan.oa.upm.filter.AjaxAuthFilter">

</bean>

<!-- mtFilter -->

<bean id="mtFilter" class="com.sankuai.meituan.filter.spring.FilterFactoryBean">

<property name="appkey" value="mtoa" />

<property name="loginUrl" value="http://sso.sankuai.com" />

<property name="authUrl" value="http://upm.sankuai.com" />

<property name="successUrl" value="/" />

<property name="filters">

<map>

<entry key="ajaxauth" value-ref="ajaxAuthFilter" />

</map>

</property>

<property name="filterChainDefinitions">

<value>

/ = anon

/static/\*\* = anon

/oa/xm/\*\* = anon

/localhost/\*\* = anon

/api/\*\* = user

/service/\*\* = user

/oa/pf/mds/dbconsole/\*\* = ajaxauth

/oa/pf/bpm/oa.pf.bpm.task.commit.ext.action = ajaxauth

/\*\* = user

</value>

</property>

</bean>

**//将applicationContext.xml中的namespace从3.0升级到3.1.xsd， 然后就可以在beans加入不同环境的定义，在此定义了pro（产品）和dev（开发），web.xml**

<beans profile="prod">

**//property-placeholder引入properties文件，所有都要加ignore-unresolvable=”true”，避免多个<context:property-placeholder>报“Could not resolve placeholder”错误**

<context:property-placeholder

ignore-unresolvable="true" location="classpath:mt.oa.properties"/>

<context:property-placeholder

ignore-unresolvable="true" location="classpath:mt.oa.data.properties" />

<context:property-placeholder

ignore-unresolvable="true" location="classpath:mt.oa.service.properties" />

**<!-- 数据源配置, 使用DBCP数据库连接池 -->**

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

destroy-method="close">

<!-- Connection Info -->

<property name="driverClassName" value="${jdbc.driver}" />

<property name="url" value="${jdbc.url}" />

<property name="username" value="${jdbc.username}" />

<property name="password" value="${jdbc.password}" />

<!-- Connection Pooling Info -->

<property name="maxActive" value="${dbcp.maxActive}" />

<property name="maxIdle" value="${dbcp.maxIdle}" />

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

**<!-- 连接Idle一个小时后超时 -->**

<property name="timeBetweenEvictionRunsMillis" value="3600000" />

<property name="minEvictableIdleTimeMillis" value="3600000" />

</bean>

**//引入其他配置文件**

<import resource="classpath:/spring/mt.oa.repository.spring.xml" />

<import resource="classpath:/spring/mt.oa.mongodb.xml" />

<import resource="classpath:spring/activiti.cfg.xml" />

<!--

<import resource="classpath:spring/mt.oa.schedule.xml"/>

<import resource="classpath:spring/mt.oa.cache.xml"/>

-->

</beans>

<beans profile="dev">

<context:property-placeholder

ignore-unresolvable="true" location="classpath:mt.oa.dev.properties" />

<context:property-placeholder

ignore-unresolvable="true" location="classpath:mt.oa.dev.data.properties" />

<context:property-placeholder

ignore-unresolvable="true" location="classpath:mt.oa.dev.service.properties" />

**<!-- 数据源配置, 使用DBCP数据库连接池 -->**

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

destroy-method="close">

<!-- Connection Info -->

<property name="driverClassName" value="${jdbc.driver}" />

<property name="url" value="${jdbc.url}" />

<property name="username" value="${jdbc.username}" />

<property name="password" value="${jdbc.password}" />

<!-- Connection Pooling Info -->

<property name="maxActive" value="${dbcp.maxActive}" />

<property name="maxIdle" value="${dbcp.maxIdle}" />

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

<!-- 连接Idle一个小时后超时 -->

<property name="timeBetweenEvictionRunsMillis" value="3600000" />

<property name="minEvictableIdleTimeMillis" value="3600000" />

</bean>

<import resource="classpath:/spring/mt.oa.repository.spring.xml" />

<import resource="classpath:/spring/mt.oa.mongodb.xml" />

<import resource="classpath:spring/activiti.cfg.xml" />

<!--

<import resource="classpath:spring/mt.oa.schedule.xml"/>

<import resource="classpath:spring/mt.oa.cache.xml"/>

-->

</beans>

</beans>

## mt.oa.mvc.xml 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:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd">

**<!-- 自动扫描且只扫描@Controller和@ControllerAdvice -->**

<context:component-scan base-package="com.meituan.oa" 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>

<!-- 注册给mvc用的 ？？？-->

<bean id="json\_convert\_message" class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter" />

//**内容协商管理器是由工厂类制造，须配置mediaTypes，如果请求url后缀为json，如**[**http://localhost/xxx/data.json**](http://localhost/xxx/data.json)**，则响应内容设置为application/json**

<bean id="contentNegotiationManager" class="org.springframework.web.accept.ContentNegotiationManagerFactoryBean">

<property name="mediaTypes">

<value>

json=application/json

xml=application/xml

\*=\*/\*

</value>

</property>

</bean>

**<!-- mvc:annotation-driven：注解映射支持**

**content-negotiation-manager指定内容协商管理器的bean**

**？？？-->**

<mvc:annotation-driven content-negotiation-manager="contentNegotiationManager">

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

<bean class="com.meituan.oa.pf.mvc.convert.OAHttpMessageConvert"></bean>

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

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

</bean>

</mvc:message-converters>

<mvc:argument-resolvers>

<bean class="com.meituan.oa.pf.mvc.bind.support.ClientEnvArgumentResolver" />

<bean

class="com.meituan.oa.pf.mvc.bind.support.GenericVOArgumentResolver" />

</mvc:argument-resolvers>

</mvc:annotation-driven>

**<!—mvc:interceptors：为MVC注入拦截器**

**此处注册LogInterceptor日志拦截器，对/oa/,/api/,/service/后的所有请求拦截-->**

<mvc:interceptors>

<mvc:interceptor>

<mvc:mapping path="/oa/\*\*"/>

<mvc:mapping path="/api/\*\*"/>

<mvc:mapping path="/service/\*\*"/>

<bean class="com.meituan.oa.pf.mvc.interceptor.LogInterceptor"/>

</mvc:interceptor>

</mvc:interceptors>

**//引入两个properties文件**

<context:property-placeholder ignore-unresolvable="true" location="classpath:mt.oa.properties" />

<context:property-placeholder ignore-unresolvable="true" location="classpath:mt.oa.data.properties" />

**// CommonsMultipartResolver上传文件，maxUploadSize最大上传大小**

<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

<property name="maxUploadSize" value="10000000" />

</bean>

**<!--InternalResourceViewResolver对Controller返回的ModelAndView前缀加 /WEB-INF/views/，后缀加.ftl，没用到-->**

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

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

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

</bean>

**<!--**

**配置<mvc:default-servlet-handler />后，会在Spring MVC上下文中定义一个org.springframework.web.servlet.resource.**

**DefaultServletHttpRequestHandler，它会像一个检查员，对进入DispatcherServlet的URL进行筛查，如果发现是静态资源的请求，就将该请求转由Web应用服务器默认的Servlet处理，如果不是静态资源的请求，才由DispatcherServlet继续处理。**

**-->**

<mvc:default-servlet-handler />

**<!--**

**mvc:view-controller如果当前路径是/，则不通过Controller处理，直接映射为index，再根据InternalResourceViewResolver，得到的视图是 /WEB-INF/views/ index.ftl**

**-->**

<mvc:view-controller path="/" view-name="index" />

<mvc:view-controller path="/web/mashup-client" view-name="/web/mashup-client" />

**<!—**

**Spring3.0中对异常的处理方法一共提供了两种：一种是使用HandlerExceptionResolver接口；一种是在Controller类内部使用@ExceptionHandler注解。使用第一种方式可以实现全局异常控制，并且Spring已经提供了一个默认的实现类SimpleMappingExceptionResolver；使用第二种方式可以在Controller内部实现更个性化点异常处理方式，灵活性更高。**

**此处定义全局异常处理，exceptionMapping异常处理映射，key异常类，value异常处理视图。**

**BaseController 定义所有Controller的异常处理方法。**

**-->**

<bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">

<property name="exceptionMappings">

<props>

<prop key="java.lang.Throwable">error/500</prop>

</props>

</property>

</bean>

</beans>

## mt.oa.repository.spring.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:jdbc="http://www.springframework.org/schema/jdbc"

xmlns:jee="http://www.springframework.org/schema/jee"

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-3.2.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd

http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd

http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.2.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd"

default-lazy-init="true">

<description>持久层配置 </description>

//扫描@Repository和@Service，禁止扫描默认

<context:component-scan base-package="com.meituan.oa.pf.repository" use-default-filters="false">

<context:include-filter type="annotation" expression="org.springframework.stereotype.Repository"/>

</context:component-scan>

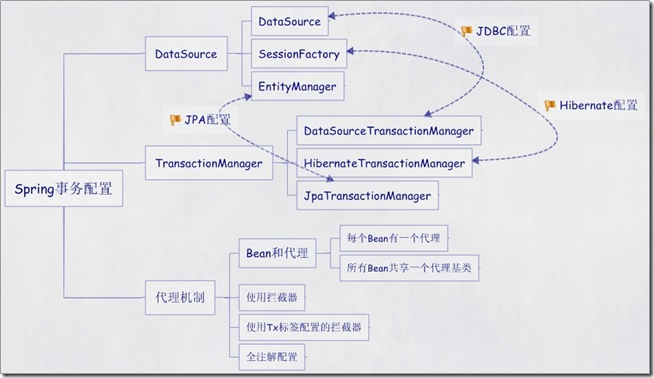
<context:component-scan base-package="com.meituan.oa.pf" use-default-filters="false">

<context:include-filter type="annotation" expression="org.springframework.stereotype.Service"/>

</context:component-scan>

<!-- 数据链接提供者，没用到 -->

<bean id="db.provider.normal" class="com.meituan.oa.pf.repository.service.impl.DBConnectionProviderServiceImpl"></bean>



**<!—配置事务通知，transaction-manager属性指定事务管理器，< tx:attributes >指定具体需要拦截的方法-->**

<tx:advice id="txAdvice" transaction-manager="transactionManager">

<tx:attributes>

<!—tx:method：拦截以get开头的方法，事务只读，事务传播行为为SUPPORTS

* REQUIRED：支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择。
* SUPPORTS：支持当前事务，如果当前没有事务，就以非事务方式执行。
* MANDATORY：支持当前事务，如果当前没有事务，就抛出异常。
* REQUIRES\_NEW：新建事务，如果当前存在事务，把当前事务挂起。
* NOT\_SUPPORTED：以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。
* NEVER：以非事务方式执行，如果当前存在事务，则抛出异常。
* NESTED：支持当前事务，如果当前事务存在，则执行一个嵌套事务，如果当前没有事务，就新建一个事务。

-->

<tx:method name="get\*" read-only="true" propagation="SUPPORTS"/>

<tx:method name="query\*" read-only="true" propagation="SUPPORTS"/>

<tx:method name="fetch\*" read-only="true" propagation="SUPPORTS"/>

<tx:method name="find\*" read-only="true" propagation="SUPPORTS"/>

<tx:method name="\*" propagation="REQUIRED" rollback-for="com.meituan.oa.pf.exception.BuzException"/>

</tx:attributes>

</tx:advice>

**<!—AOP配置**

**(\* com.meituan.oa..service..\*BizService\*.\*(..))**

**第一个\*：任意返回值**

**第二个..：com.meituan.oa包即其子包下的。**

**第三个..：service包及其子包下的。**

**第四个\*BizService\*：包含BizService的类**

**第无个\*：任意方法名**

**第六个(..)：包含0个或多个参数**

**-->**

<aop:config>

**<!--  声明事务切入点(配置哪些类的哪些方法参与事务) -->**

<aop:pointcut id="com.meituan.oa" expression="execution(\* com.meituan.oa..service..\*BizService\*.\*(..))" />

**<!-- 通知器(把事务通知绑定到切入点) -->**

<aop:advisor advice-ref="txAdvice" pointcut-ref="com.meituan.oa" />

</aop:config>

</beans>

## mt.oa.mongodb.xml

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mongo="http://www.springframework.org/schema/data/mongo"

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

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd

http://www.springframework.org/schema/data/mongo http://www.springframework.org/schema/data/mongo/spring-mongo-1.0.xsd">

<context:property-placeholder ignore-unresolvable="true" location="classpath:mt.oa.data.properties" />

<mongo:db-factory dbname="mt\_oa\_pf"

host="${mongodb.host}"

port="${mongodb.port}"/>

<bean id="mongoTemplate" class="org.springframework.data.mongodb.core.MongoTemplate">

<constructor-arg ref="mongoDbFactory"/>

</bean>

</beans>

## activity.cfg.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"

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

<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="mailServerHost" value="localhost" />

<property name="mailServerPort" value="5025" />

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

</bean>

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

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

</bean>

<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>

## mt.oa.schedule.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:jdbc="http://www.springframework.org/schema/jdbc"

xmlns:jee="http://www.springframework.org/schema/jee"

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-3.2.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd

http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd

http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.2.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd"

default-lazy-init="true">

<description>任务调度</description>

<bean id="spring.scheduler" class="org.springframework.scheduling.quartz.SchedulerFactoryBean"

lazy-init="false" destroy-method="destroy">

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

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

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

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

<property name="applicationContextSchedulerContextKey" value="appctx"/>

<property name="configLocation" value="classpath:quartz.properties"/>

<property name="startupDelay" value="30"/><!-- 延时启动 -->

<!-- <property name="overwriteExistingJobs" value="true"/> --><!-- 启动时更新己存在的Job，这样就不用每次修改targetObject后删除qrtz\_job\_details表对应记录了 -->

</bean>

</beans>

# 测试配置文件

## mt.oa.test.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:jdbc="http://www.springframework.org/schema/jdbc" xmlns:jee="http://www.springframework.org/schema/jee"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd

http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd

http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.2.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd"

default-lazy-init="true">

<description>Spring公共配置 </description>

**//扫描@Controller，@Service，@Component**

<context:component-scan base-package="com.meituan.oa">

<context:include-filter type="annotation"

expression="org.springframework.stereotype.Controller" />

<context:include-filter type="annotation"

expression="org.springframework.stereotype.Service" />

<context:include-filter type="annotation"

expression="org.springframework.stereotype.Component" />

</context:component-scan>

**//引入事务管理**

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

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

</bean>

**//引入properties文件**

<context:property-placeholder ignore-unresolvable="true" location="classpath:mt.oa.dev.properties"/>

<context:property-placeholder ignore-unresolvable="true" location="classpath:mt.oa.dev.data.properties" />

<context:property-placeholder ignore-unresolvable="true" location="classpath:mt.oa.dev.service.properties"/>

**<!-- 数据源配置, 使用DBCP数据库连接池 -->**

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

destroy-method="close">

<!-- Connection Info -->

<property name="driverClassName" value="${jdbc.driver}" />

<property name="url" value="${jdbc.url}" />

<property name="username" value="${jdbc.username}" />

<property name="password" value="${jdbc.password}" />

<!-- Connection Pooling Info -->

<property name="maxActive" value="${dbcp.maxActive}" />

<property name="maxIdle" value="${dbcp.maxIdle}" />

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

<!-- 连接Idle一个小时后超时 -->

<property name="timeBetweenEvictionRunsMillis" value="3600000" />

<property name="minEvictableIdleTimeMillis" value="3600000" />

</bean>

**//配置邮件服务**

<bean id="mail.send" class="org.springframework.mail.javamail.JavaMailSenderImpl">

<!-- SMTP settings -->

<property name="host" value="${smtp\_host}" />

<property name="port" value="${smtp\_port}" />

<property name="username" value="${mail\_user\_name}" />

<property name="password" value="${mail\_pwd}" />

<property name="javaMailProperties">

<!-- additional properties specific to JavaMail -->

<props>

<prop key="mail.transport.protocol">smtp</prop>

<prop key="mail.smtp.auth">true</prop>

<prop key="mail.smtp.starttls.enable">true</prop>

</props>

</property>

</bean>

<import resource="classpath:/spring/mt.oa.repository.spring.xml"/>

<import resource="classpath:/spring/mt.oa.mongodb.xml"/>

<import resource="classpath:/spring/activiti.cfg.xml"/>

<!-- <import resource="classpath:/spring/mt.oa.schedule.xml"/> -->

<!-- <import resource="classpath:spring/mt.oa.cache.xml"/> -->

<!-- <import resource="classpath:spring/mt.oa.ldap.xml"/> -->

</beans>

# 事务管理

<http://www.ibm.com/developerworks/cn/java/j-lo-spring-ts1/>

<http://www.ibm.com/developerworks/cn/education/opensource/os-cn-spring-trans/section2.html>

问题：

<http://192.168.2.165:8080/#/account/login如何映射的>到account/login.js？

返回的ModelAndView根据配置应该/WEB-INF/views/xxx.ftl，如何映射到static/下的js文件

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

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

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

</bean>