SpringMVC配置

web.xml

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>springmvcProject</display-name>

    <!--配置springmvc的核心控制器-->
  <servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <!--配置初始化参数，用于读取springmvc的配置文件-->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath*:springmvc.xml</param-value>
    </init-param>
      <!--配置servlet的对象创建点：应用加载时创建，取值只能是非0正整数，表示启动顺序-->
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <!--任何请求都过这个servlet-->
    <url-pattern>/</url-pattern>
  </servlet-mapping>


    <!--配置springMVC编码过滤器-->
    <filter>
        <filter-name>CharacterEncodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
     <!--设置过滤器中的属性值-->
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
        <!--启动过滤器-->
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <!--过滤所有请求-->
    <filter-mapping>
        <filter-name>CharacterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!--整合Spring和SpringMVC-->
```

```
    <!--配置Spring提供的监听器，用于启动服务时加载容器 该监听器只能加载名称为application.xml的
配置文件-->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
    </listener>
    <!--手动指定spring配置文件位置-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath*:application.xml</param-value>
    </context-param>

    <!--启动页面-->
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

springmvc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       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-3.0.xsd
     http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
     "
       default-lazy-init="true">

<!--开启注解扫描，创建spring容器要扫描的包-->
    <context:component-scan base-package="com.hh">
        <!--制定扫描规则，只扫描使用@Controller注解的Java类-->
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!--配置视图解析器-->
    <bean id="internalResourceViewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/pages/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

    <!--配置自定义类型转换器-->
    <bean id ="conversionService"
class="org.springframework.context.support.ConversionServiceFactoryBean">

        <property name="converters">
            <set>
```

```
                <bean class="com.hh.utils.StringToDateConverter"/>
            </set>
        </property>
    </bean>

    <!--开启SpringMVC框架注解支持-->
<!--自定义类型转换器需要显示添加注解-->
    <mvc:annotation-driven conversion-service="conversionService"/>

    <!--在springmvc的配置文件中可以配置，静态资源不过滤-->
    <!--location表示路径，mapping表示文件,**表示该目录下的文件及其子目录的文件-->
    <mvc:resources mapping="/css" location="/css/**"/>

    <!--文件上传配置  id是固定值-->
    <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="maxUploadSize">
            <value>5242880</value>
        </property>
    </bean>
</beans>
```

Mybatis配置数据

举个例子

```
<!--连接数据源-->
<!--采用服务器提供的JNDI技术实现来获取DataSource对象，不同的服务器所能拿到的dataSource是不一样的
 注意：如果不是web或者maven的war工程，是不能使用。
 此处是用的resin服务器，使用了jndi
-->
    <bean id="dataSourceVVdmm" class="org.springframework.jndi.JndiObjectFactoryBean">
        <property name="jndiName" value="java:comp/env/jdbc/vv_dmm"/>
    </bean>
    <!--配置Mybatis的Session工厂-->
    <bean id="sqlSessionFactoryVVdmm" class="org.mybatis.spring.SqlSessionFactoryBean">
        <!--数据池连接池-->
        <property name="dataSource" ref="dataSourceVVdmm"/>
        <!--加载mybatis的全局配置文件-->
        <property name="mapperLocations" value="classpath:com/vv/dmm/mapper/*.xml"/>
    </bean>
    <!--配置Mapper扫描器-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.vv.dmm.mapper"/>
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactoryVVdmm"/>
    </bean>
    <!--配置事务管理器-->
    <bean id="transactionManagerVVdmm"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSourceVVdmm"/>
    </bean>
    <!--开启事务-->
    <tx:annotation-driven transaction-manager="transactionManagerVVdmm"/>
```