

一 Spring 配置

1 在普通的 JAVA 项目中，获取由 Spring 管理的对象分为几步

```
//1、初始化容器
ApplicationContext context=new
ClassPathXmlApplicationContext("applicationContextZhuJie.xml");
//2、获取 User 对象
User user = (User)context.getBean("user");
//3、相关操作

//4、关闭容器
((ClassPathXmlApplicationContext)context).close();
```

2 在 WEB 项目中，Spring 应该在什么时候初始化呢？

因为 Spring 是全局使用，应该在 WEB 应用初始化时初始化容器。

3 那么应该怎样在 WEB 应用初始化时初始化 Spring？

显然只有通过监听器，监听 ServletContext 初始化，在 WEB 应用初始化时初始化容器

4 开始配置

Spring 已经给我们提供了这个监听器（ContextLoaderListener），该监听器实现了原生的 ServletContextListener 接口。

4.A 创建 Spring 配置文件 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:util="http://www.springframework.org/schema/util"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
```

```

        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/aop
https://www.springframework.org/schema/aop/spring-aop.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <!-- 开启 IOC 扫描包 -->
    <context:component-scan base-package="cn.shu.blog.beans"/>
    <context:component-scan base-package="cn.shu.blog.service"/>
    <context:component-scan base-package="cn.shu.blog.dao"/>
    <context:component-scan base-package="cn.shu.blog.aspect"/>

    <!-- 开启注解配置 DI -->
    <context:annotation-config/>

    <!-- 引入 properties 文件 -->
    <context:property-placeholder location="classpath:/user.properties"/>

    <!-- 开启注解方式 AOP proxy-target-class="true" 表示使用 cglib 代理模式 -->
    <aop:aspectj-autoproxy proxy-target-class="true"/>

</beans>

```

4.B 在 WEB.XML 中配置监听器

```

<!-- ##### Spring 监听器, 初始化 Spring ##### -->
<listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
</listener>
<!-- Spring 初始化时会加载此参数 指定 Spring 的配置文件路径 -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
</context-param>

```

这样配置完成后，在 WEB 应用启动时，就可以初始化 Spring 容器，需要用的地址可以通过注解注入对象。

二 Spring MVC 配置

基本上没什么变化，按 Spring MVC 标准配置即可

1 创建 SpringMVC.xml 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/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <!--开启 Spring MVC 的包扫描-->
    <context:component-scan base-package="cn.shu.blog.controller"/>
    <!--开启注解方式 MVC-->
    <mvc:annotation-driven/>
</beans>
```

2 SpringMVC.xml 配置文件中配置视图解析器

```
<?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/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <!--开启 Spring MVC 的包扫描-->
    <context:component-scan base-package="cn.shu.blog.controller"/>
    <!--配置视图解析器-->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!--前缀和后缀，处理器适配器返回视图后会默认在视图名加上前缀后缀-->
        <property name="prefix" value="/WEB-INF/jsp"/>
        <property name="suffix" value=".jsp"/>
    </bean>
    <!--开启注解方式 MVC-->
    <mvc:annotation-driven/>
</beans>
```

3 WEB.XML 中添加前端控制器

本质就是 Servlet，前端控制器里需配置 SpringMVC 配置文件的路径

```
<!--##### 配置 SpringMVC 的前端控制器 本质为 Servlet#####-->
<servlet>
  <servlet-name>springMvc</servlet-name>
  <!--如果报错 DispatcherServlet' is not assignable to javax.Servlet 则需要
导入 tomcat-->
  <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <!--如果不配置 会到默认目录找默认文件-->
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:springMVC.xml</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>springMvc</servlet-name>
  <url-pattern>*.action</url-pattern><!--随便写个路径就可以-->
</servlet-mapping>
```

三 MyBatis 配置

MyBatis 不属于 Spring 全家桶，所以整合稍微麻烦一些，而且需要导入一些整合包
在普通项目中使用 MyBatis 步骤

```
//1、加载配置文件
InputStream inputStream =
Resources.getResourceAsStream("sqlMapConfig.xml");
```




```
//2、创建连接工厂
SqlSessionFactory build = new
SqlSessionFactoryBuilder().build(inputStream);
//3、获取连接
SqlSession sqlSession = build.openSession();
//4、获取接口的实现类
User mapper = sqlSession.getMapper(User.class);
//5、相关操作
String userName = mapper.getUserName();
//6、关闭连接
sqlSession.close();
```

如何把这个实现类交给 Spring 管理，然后全局@Autowired 调用呢？

MyBatis 提供了相关的类来加载相关配置，创建对应的实现类

1 导包

注：这里导入的是整合包，真正使用时其它相关包也需要导入

	mybatis-3.2.2.jar	2019-9-12 14:36	Executable Jar File	684 KB
	mybatis-ehcache-1.0.2.jar	2016-10-9 16:16	Executable Jar File	10 KB
	mybatis-spring-1.2.0.jar	2016-10-9 16:16	Executable Jar File	48 KB

2 创建 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>
</configuration>
```

3 创建 SQL 映射文件

省略

4 在 Spring 配置文件中配置(整合到 Spring)

```
<!--#####配置 c3p0 的数据源#####-->
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="com.mysql.jdbc.Driver"/>
    <property name="jdbcUrl" value="jdbc:mysql:///myblog"/>
    <property name="user" value="root"/>
    <property name="password" value="admin"/>
</bean>

<!--#####整合 Mybatis#####-->
<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
    <!--数据源-->
    <property name="dataSource" ref="dataSource"/>
    <!--MyBatis 配置文件-->
    <property name="configLocation" value="classpath:/sqlMapConfig.xml"/>
    <!--sql 配置文件-->
    <property name="mapperLocations" value="classpath:/mappers/*.xml"/>
</bean>

<!--#####配置 MyBatis 的 bean 生成器 #####-->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="cn.shu.blog.dao"/>
</bean>
```

5 配置事务管理(Spring 配置文件中)

```
<!--#####配置事务管理器#####-->
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <!-- 指定数据源 -->
    <property name="dataSource" ref="dataSource"/>
</bean>

<!--#####开启注解方式配置事务#####-->
<tx:annotation-driven transaction-manager="transactionManager"/>
```