1. 配置文件
2. **web.xml**

<!-- Spring MVC配置 -->  
<servlet>  
 <servlet-name>spring</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
  
 <!--可以自定义servlet.xml配置文件的位置和名称，默认为WEB-INF目录下，名称为[<servlet-name>]-servlet.xml，如spring-servlet.xml-->  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring-servlet.xml</param-value>  
 </init-param>-->  
  
 <!--load-on-startup元素标记容器是否在启动的时候就加载这个servlet(实例化并调用其init()方法) -->  
 <load-on-startup>1</load-on-startup>  
</servlet>  
  
<servlet-mapping>  
 <servlet-name>spring</servlet-name>  
 <!--<url-pattern>\*.html</url-pattern>-->  
 <url-pattern>\*.action</url-pattern>  
</servlet-mapping>

<url-pattern>\*.**action**</url-pattern> 会拦截\*.**action**结尾的请求。

<servlet-name>spring</servlet-name>这个Servlet的名字是spring，可以有多个DispatcherServlet，是通过名字来区分的。每一个DispatcherServlet有自己的WebApplicationContext上下文对象。

在DispatcherServlet的初始化过程中，框架会在web应用的 WEB-INF文件夹下寻找名为[dispatcherServlet]-servlet.xml 的配置文件，生成文件中定义的bean。文件路径也可以自己指定，如下：

    <init-param>

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

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

    </init-param>

指明了配置文件的文件名，不使用默认配置文件名，而使用springMVC.xml配置文件。

其中<param-value>\*\*.xml</param-value> 这里可以使用多种写法

1、不写,使用默认值:/WEB-INF/<servlet-name>-servlet.xml

2、<param-value>/WEB-INF/classes/springMVC.xml</param-value>

3、<param-value>classpath\*:springMVC-mvc.xml</param-value>

4、多个值用逗号分隔

1. **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: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-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/mvc  
 http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd"**>  
  
 <!-- 启动注解驱动的Spring MVC功能，注册请求url和注解POJO类方法的映射-->  
 <mvc:annotation-driven >  
  
 </mvc:annotation-driven>  
  
 <!-- 启动包扫描功能，以便注册带有@Controller、@service、@repository、@Component等注解的类成为spring的bean -->  
 <context:component-scan base-package**="mars.controller"** />  
 <!-- 对模型视图名称的解析，在请求时模型视图名称添加前后缀 -->  
 <bean class**="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 <property name**="viewClass"** value**="org.springframework.web.servlet.view.JstlView"**/>  
 <property name**="prefix"** value**="/"**/> <!-- 前缀 -->  
 <property name**="suffix"** value**=".jsp"**/> <!-- 后缀 -->  
 </bean>  
 <!--访问静态文件（jpg,js,css）的方法-->  
 <mvc:resources location**="/files/"** mapping**="/files/\*"** />  
 <mvc:resources location**="/scripts/"** mapping**="/scripts/\*"** />  
 <mvc:resources location**="/styles/"** mapping**="/styles/\*"** />  
 <mvc:resources location**="/views/"** mapping**="/views/\*"** />  
</beans>

1. 注解

1.@Controller定义一个控制器

2.@RequestMapping映射请求

3.@RequestParam绑定请求参数到方法参数

示例：

**@RequestMapping**(value = **"/addProduct"**, method = RequestMethod.*POST*)  
**@ResponseBody**public JSONObject addProduct(**@RequestBody** String body){

JSONObject result = new JSONObject();

return result;  
}

4、@ModelAttribute和 @SessionAttributes

@ModelAttribute代表的是：该Controller的所有方法在调用前，先执行此@ModelAttribute方法，可用于注解和方法参数中，可以把这个@ModelAttribute特性，应用在BaseController当中，所有的Controller继承BaseController，即可实现在调用Controller时，先执行@ModelAttribute方法。

@SessionAttributes即将值放到session作用域中，写在class上面。

5.@PathVariable

用于将请求URL中的模板变量映射到功能处理方法的参数上，即取出uri模板中的变量作为参数。如：

**@RequestMapping**(value=**"/product/{productId}"**,method = RequestMethod.*GET*)  
public String getProduct(**@PathVariable**(**"productId"**) String productId){  
 System.*out*.println(**"Product Id : "** + productId);  
 return **"hello"**;  
}

6.@Repository

用于注解dao层，在daoImpl类上面注解。

7.@ResponseBody

作用： 该注解用于将Controller的方法返回的对象，通过适当的HttpMessageConverter转换为指定格式后，写入到Response对象的body数据区。

使用时机：返回的数据不是html标签的页面，而是其他某种格式的数据时（如json、xml等）使用；

参考：http://www.cnblogs.com/leskang/p/5445698.html