静态资源访问（jsp,css,images.....）

一，SpringMVC静态资源拦截问题

SpringMVC的工作机制是：来自浏览器的所有访问都会被前端控制器（DispatcherServlet）捕获，然后前端控制器把请求转交给处理器映射（HandlerMapping）,HandlerMapping为请求分配对应的控制器（Controller）进行请求处理。

```
<servlet>
      <servlet-name>springMVC</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>springMVC</servlet-name>
      <url-pattern>/</url-pattern>
   </servlet-mapping>
```

通过上面的配置，DispatcherServlet将捕获Web容器所有请求，包括静态资源请求。

二，解决静态资源拦截问题

方案一：拦截器中增加针对静态资源不进行过滤(涉及spring-mvc.xml)

```
<mvc:annotation-driven/>
<!--
   通过mvc:resources设置静态资源，这样servlet就会处理这些静态资源，而不通过控制器设置不过滤内容，比如:css,js,img 等资源文件
   location指的是本地的真实路径，mapping指的是映射到的虚拟路径。
-->
<mvc:resources location="/js/" mapping="/js/**"/>
```

方案二：使用默认的servlet处理静态资源(涉及spring-mvc.xml，web.xml)

在spring-mvc.xml中添加：

```
<!--增加对静态资源的处理,当前的设置必须在Spring的Dispatcher的前面-->
 <servlet-mapping>
```

```
    <servlet-name>default</servlet-name>
    <url-pattern>*.css</url-pattern>
    <url-pattern>/css/*</url-pattern>
  </servlet-mapping>
```

方案三：修改spring的全局拦截设置为*.do的拦截(涉及web.xml)

```
 <!-- 拦截所有请求 -->
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <!--<url-pattern>/</url-pattern>-->
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
```

比较总结：

第一种方案配置比较臃肿，多个拦截器时增加文件行数，不推荐使用

第二种方案使用默认的Servlet进行资源文件的访问，Spring拦截所有请求，然后再将资源文件交由默认的Sevlet进行处理，性能上少有损耗

第三种方案Spring只是处理以'.do'结尾的访问，性能上更加高效，但是再访问路径上必须都以'.do'结尾，URL不太文雅

综上所述，推荐使用第二和第三种方案