**Spring MVC 使用kaptcha生成验证码**

1.下载kaptcha-2.3.2.jar(或直接通过该文章附件下载)

<http://code.google.com/p/kaptcha/downloads/list>

kaptcha-2.3.2-jdk14.jar

kaptcha-2.3.2.jar

1. 向spring.xml中添加bean

<bean id=*"captchaProducer"* class=*"com.google.code.kaptcha.impl.DefaultKaptcha"*>

<property name=*"config"*>

<bean class=*"com.google.code.kaptcha.util.Config"*>

<constructor-arg>

<props>

<prop key=*"kaptcha.border"*>yes</prop>

<prop key=*"kaptcha.border.color"*>105,179,90</prop>

<prop key=*"kaptcha.textproducer.font.color"*>blue</prop>

<prop key=*"kaptcha.image.width"*>125</prop>

<prop key=*"kaptcha.image.height"*>45</prop>

<prop key=*"kaptcha.textproducer.font.size"*>40</prop>

<prop key=*"kaptcha.session.key"*>code</prop>

<prop key=*"kaptcha.textproducer.char.length"*>4</prop>

<prop key=*"kaptcha.textproducer.font.names"*>宋体,楷体,微软雅黑 </prop>

</props>

</constructor-arg>

</bean>

</property>

</bean>

1. 编写controller（设置）

@Controller

@RequestMapping("/yzm/")

**public** **class** YZMController {

@Autowired

**private** Producer producer;

@RequestMapping("/yzm")

**public** **void** yzm(HttpServletRequest request, HttpServletResponse response,

HttpSession session) **throws** IOException {

String lastYZM = (String) session.getAttribute("yzm");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*上一次的验证码是" + lastYZM+ "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

// 设置浏览器不要对数据进行缓存

response.setDateHeader("Expires", 0);

// Set standard HTTP/1.1 no-cache headers.

response.setHeader("Cache-Control",

"no-store, no-cache, must-revalidate");

// Set IE extended HTTP/1.1 no-cache headers (use addHeader).

response.addHeader("Cache-Control", "post-check=0, pre-check=0");

// Set standard HTTP/1.0 no-cache header.

response.setHeader("Pragma", "no-cache");

// return a jpeg

response.setContentType("image/jpeg");

String yzm = producer.createText();

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*本次的验证码是" + yzm+ "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

session.setAttribute("yzm", yzm);

BufferedImage image = producer.createImage(yzm);

OutputStream out = response.getOutputStream();

ImageIO.*write*(image, "jpg", out);

**try** {

out.flush();

} **finally** {

out.close();

}

}

}

1. kaptcha可配置选项

kaptcha.border 是否有边框 默认为true 我们可以自己设置yes，no

kaptcha.border.color 边框颜色 默认为Color.BLACK

kaptcha.border.thickness 边框粗细度 默认为1

kaptcha.producer.impl 验证码生成器 默认为DefaultKaptcha

kaptcha.textproducer.impl 验证码文本生成器 默DefaultTextCreator

kaptcha.textproducer.char.string 验证码文本字符内容范围 默认为abcde2345678gfynmnpwx

kaptcha.textproducer.char.length 验证码文本字符长度 默认为5

kaptcha.textproducer.font.names 验证码文本字体样式 默认为new Font("Arial", 1, fontSize), new Font("Courier", 1, fontSize)

kaptcha.textproducer.font.size 验证码文本字符大小 默认为40

kaptcha.textproducer.font.color 验证码文本字符颜色 默认为Color.BLACK

kaptcha.textproducer.char.space 验证码文本字符间距 默认为2

kaptcha.noise.impl 验证码噪点生成对象 默认为DefaultNoise

kaptcha.noise.color 验证码噪点颜色 默认为Color.BLACK

kaptcha.obscurificator.impl 验证码样式引擎 默认为WaterRipple

kaptcha.word.impl 验证码文本字符渲染 默认为DefaultWordRenderer

kaptcha.background.impl 验证码背景生成器 默认为DefaultBackground

kaptcha.background.clear.from 验证码背景颜色渐进 默认为Color.LIGHT\_GRAY

kaptcha.background.clear.to 验证码背景颜色渐进 默认为Color.WHITE

kaptcha.image.width 验证码图片宽度 默认为200

kaptcha.image.height 验证码图片高度 默认为50

|  |  |  |
| --- | --- | --- |
| ****Constant**** | ****描述**** | ****默认值**** |
| kaptcha.border | 图片边框，合法值：yes , no | yes |
| kaptcha.border.color | 边框颜色，合法值： r,g,b (and optional alpha) 或者 white,black,blue. | black |
| kaptcha.border.thickness | 边框厚度，合法值：>0 | 1 |
| kaptcha.image.width | 图片宽 | 200 |
| kaptcha.image.height | 图片高 | 50 |
| kaptcha.producer.impl | 图片实现类 | com.google.code.kaptcha.impl.DefaultKaptcha |
| kaptcha.textproducer.impl | 文本实现类 | com.google.code.kaptcha.text.impl.DefaultTextCreator |
| kaptcha.textproducer.char.string | 文本集合，验证码值从此集合中获取 | abcde2345678gfynmnpwx |
| kaptcha.textproducer.char.length | 验证码长度 | 5 |
| kaptcha.textproducer.font.names | 字体 | Arial, Courier |
| kaptcha.textproducer.font.size | 字体大小 | 40px. |
| kaptcha.textproducer.font.color | 字体颜色，合法值： r,g,b  或者 white,black,blue. | black |
| kaptcha.textproducer.char.space | 文字间隔 | 2 |
| kaptcha.noise.impl | 干扰实现类 | com.google.code.kaptcha.impl.DefaultNoise |
| kaptcha.noise.color | 干扰 颜色，合法值： r,g,b 或者 white,black,blue. | black |
| kaptcha.obscurificator.impl | 图片样式：  水纹com.google.code.kaptcha.impl.WaterRipple  鱼眼com.google.code.kaptcha.impl.FishEyeGimpy  阴影com.google.code.kaptcha.impl.ShadowGimpy | com.google.code.kaptcha.impl.WaterRipple |
| kaptcha.background.impl | 背景实现类 | com.google.code.kaptcha.impl.DefaultBackground |
| kaptcha.background.clear.from | 背景颜色渐变，开始颜色 | light grey |
| kaptcha.background.clear.to | 背景颜色渐变， 结束颜色 | white |
| kaptcha.word.impl | 文字渲染器 | com.google.code.kaptcha.text.impl.DefaultWordRenderer |
| kaptcha.session.key | session key | KAPTCHA\_SESSION\_KEY |
| kaptcha.session.date | session date | KAPTCHA\_SESSION\_DATE |

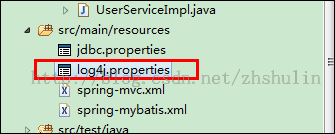
**SSM（Spring+SpringMVC+mybatis）使用Log4j记录日志**

为了方便调试，一般都会使用日志来输出信息，Log4j是[Apache](http://baike.baidu.com/subview/28283/5418753.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)的一个开放源代码项目，通过使用Log4j，我们可以控制[日志](http://baike.baidu.com/view/269351.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)信息输送的目的地是[控制台](http://baike.baidu.com/view/135167.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)、文件、[GUI](http://baike.baidu.com/view/25309.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)组件，甚至是套接口服务器、[NT](http://baike.baidu.com/subview/378/5113303.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)的事件记录器、[UNIX](http://baike.baidu.com/view/8095.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank) [Syslog](http://baike.baidu.com/view/1614723.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)[守护进程](http://baike.baidu.com/view/53123.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。

      Log4j的配置很简单，而且也是通用的，下面给出一个基本的配置，换到其他项目中也无需做多大的调整，如果想做调整或者想了解Log4j的各种配置，参看我转载的一篇博文，很详细：

[http://blog.csdn.net/zhshulin/article/details/37937365](http://blog.csdn.net/zhshulin/article/details/37937365" \t "http://blog.csdn.net/gebitan505/article/details/_blank)

下面给出配置文件目录：



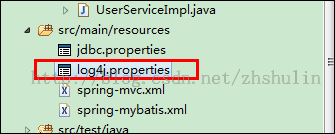
log4j.properties

为了方便调试，一般都会使用日志来输出信息，Log4j是[Apache](http://baike.baidu.com/subview/28283/5418753.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)的一个开放源代码项目，通过使用Log4j，我们可以控制[日志](http://baike.baidu.com/view/269351.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)信息输送的目的地是[控制台](http://baike.baidu.com/view/135167.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)、文件、[GUI](http://baike.baidu.com/view/25309.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)组件，甚至是套接口服务器、[NT](http://baike.baidu.com/subview/378/5113303.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)的事件记录器、[UNIX](http://baike.baidu.com/view/8095.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank) [Syslog](http://baike.baidu.com/view/1614723.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)[守护进程](http://baike.baidu.com/view/53123.htm" \t "http://blog.csdn.net/gebitan505/article/details/_blank)等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。

      Log4j的配置很简单，而且也是通用的，下面给出一个基本的配置，换到其他项目中也无需做多大的调整，如果想做调整或者想了解Log4j的各种配置，参看我转载的一篇博文，很详细：

[http://blog.csdn.net/zhshulin/article/details/37937365](http://blog.csdn.net/zhshulin/article/details/37937365" \t "http://blog.csdn.net/gebitan505/article/details/_blank)

下面给出配置文件目录：



log4j.properties#定义LOG输出级别

log4j.rootLogger=INFO,Console,File

#定义日志输出目的地为控制台

log4j.appender.Console=org.apache.log4j.ConsoleAppender

log4j.appender.Console.Target=System.out

#可以灵活地指定日志输出格式，下面一行是指定具体的格式

log4j.appender.Console.layout = org.apache.log4j.PatternLayout

log4j.appender.Console.layout.ConversionPattern=[%c] - %m%n

#文件大小到达指定尺寸的时候产生一个新的文件

log4j.appender.File = org.apache.log4j.RollingFileAppender

#指定输出目录

log4j.appender.File.File = logs/ssm.log

#定义文件最大大小

log4j.appender.File.MaxFileSize = 10MB

# 输出所以日志，如果换成DEBUG表示输出DEBUG以上级别日志

log4j.appender.File.Threshold = ALL

log4j.appender.File.layout = org.apache.log4j.PatternLayout

log4j.appender.File.layout.ConversionPattern =[%p] [%d{yyyy-MM-dd HH\:mm\:ss}][%c]%m%n