kaptcha - HowToUse.wiki

Export to GitHub

<wiki:gadget url="http://kaptcha.googlecode.com/svn/wiki/adsense3.xml" border="0" width="470" height="60" />

Introduction

<wiki:gadget url="http://kaptcha.googlecode.com/svn/wiki/donate.xml" border="0" width="150" height="75" />

Basic use of kaptcha in your webapp is quite easy. All you need to do is add the jar to your project, make a reference to the kaptcha servlet in your web.xml and then check the servlet session for the generated kaptcha value and compare it to what the user submits on your form.

For those of you looking for an audio kaptcha solution, this product probably won't ever support that unless someone else wants to work on it. Sorry.

> *Note: If you are currently using JCaptcha in the same .war file as Kaptcha, you will have a conflict with the image generation library. You must remove JCaptcha. See this issue for details.*

Example

Included in the distribution is an example .war file for you to play with. Note that this .war file will only work with Java 1.5 and higher. For easiest setup, download Tomcat. I use 5.5.x and I download the 'Core' distribution.

Then, copy the kaptcha.war file into the webapps directory. Make sure it is named 'kaptcha.war'. Then, start up Tomcat. I type on Unix: `./bin/catalina.sh run` or on Windows: `bin/catalina.bat run`.

Now, visit: http://localhost:8080/kaptcha/KaptchaExample.jsp

If you edit the web.xml file in the exploded kaptcha.war folder you can test changes to the ConfigParameters quite easily.

Details

Here are the details of how to integrate Kaptcha into your own application.

- Put the appropriate .jar file (depending on which JDK you are using) into the WEB-INF/lib directory of your .war file.

- Put the image tag into your page (checking that the path to the .jpg matches up with what is defined in your web.xml below for the url-pattern):

```

```

- Put the reference in your web.xml (checking that the url-pattern path matches up with what you put in your html fragment above):

```
<servlet> <servlet-name>Kaptcha</servlet-name> <servlet-class>com.google.code.kaptcha.servlet.KaptchaServlet</servlet-class> </servlet> <servlet-mapping> <servlet-name>Kaptcha</servlet-name> <url-pattern>/kaptcha.jpg</url-pattern> </servlet-mapping>
```

- In your code that manages the submit action:

``` String kaptchaExpected = (String)request.getSession() .getAttribute(com.google.code.kaptcha.Constants.KAPTCHA_SESSION_KEY); String kaptchaReceived = request.getParameter("kaptcha");

if (kaptchaReceived == null || !kaptchaReceived.equalsIgnoreCase(kaptchaExpected)) { setError("kaptcha", "Invalid validation code."); } ```

- Make sure to start your JDK with **-Djava.awt.headless=true**

That is it!

# Extra

If you love JQuery like I do, you can use it to easily solve the problem that happens when someone can't read the image that was generated for them. What this does is bind a function to the onclick handler for the img element so that when the image is clicked on, it will get reloaded. Note the use of the query string to append a random number to the end. This is because the browser has already cached the image and the query string makes it look like a new request.

```

$(function(){ $('#kaptchaImage').click(function () { $(this).attr('src', '/kaptcha.jpg?' + Math.floor(Math.random()*100)); }) });

Can't read the image? Click it to get a new one. ```

If you want to get fancy, you can add an extra visual effect when refreshing the image. This example uses the fadeIn() effect when you click on it:

```
$('#kaptchaImage').click(function () { $(this).hide() .attr('src', '/kaptcha.jpg?' +
Math.floor(Math.random()*100) ) .fadeIn(); })
```

Customization

There are many web.xml parameters to customize the output of the kaptcha.

<wiki:gadget url="http://kaptcha.googlecode.com/svn/wiki/donate.xml" border="0" width="150" height="75" />

<wiki:gadget url="http://kaptcha.googlecode.com/svn/wiki/adsense3.xml" border="0" width="470" height="60" />