Using reCAPTCHA with Java/JSP

The reCAPTCHA Java Library provides a simple way to place a CAPTCHA on your Java-based website, helping you stop bots from abusing it. The library wraps the reCAPTCHA API.

To use reCAPTCHA with Java/JSP, you can download the reCAPTCHA Java Library here (contributed by Soren) and unzip it. Typically the only thing you'll need is the jar file (recaptcha4j-X.X.X.jar), which you have to copy to a place where it can be loaded by your java application. For example, if you are using Tomcat to run JSP, you may put the jar file in a directory called *WEB-INF/lib/*.

Quick Start

After you've signed up for your API keys and downloaded the reCAPTCHA Java Library, below are basic instructions for installing reCAPTCHA on your site.

Client Side (How to make the CAPTCHA image show up)

If you want to use the Java plugin to display the reCAPTCHA widget, you'll need to import the appropriate reCAPTCHA classes. In JSP, you would do this by inserting these lines near the top of the file with the form element where the reCAPTCHA widget will be displayed:

```
<%@ page import="net.tanesha.recaptcha.ReCaptcha" %>
<%@ page import="net.tanesha.recaptcha.ReCaptchaFactory" %>
```

Then, you need to create an instance of reCAPTCHA:

```
ReCaptcha c = ReCaptchaFactory.newReCaptcha("your_public_key", "your_private_key", false);
```

Finally, the HTML to display the reCAPTCHA widget can be obtained from the following function call:

```
c.createRecaptchaHtml(null, null)
```

So, in JSP your code may look something like this:

```
<%@ page import="net.tanesha.recaptcha.ReCaptcha" %>
<%@ page import="net.tanesha.recaptcha.ReCaptchaFactory" %>

<html>
  <body>
    <form action="" method="post">
    <%
      ReCaptcha c = ReCaptchaFactory.newReCaptcha("your_public_key", "your_private_key", false);
      out.print(c.createRecaptchaHtml(null, null));
    %>
    <input type="submit" value="submit" />
    </form>
  </body>
</html>
```

Don't forget to replace `your_public_key` and `your_private_key` with your API key values.

Server Side (How to test if the user entered the right answer)

In the application that verifies your form, you'll first need to import the necessary reCAPTCHA classes:

```
import net.tanesha.recaptcha.ReCaptchaImpl;
import net.tanesha.recaptcha.ReCaptchaResponse;
```

Next, you need to insert the code that verifies the reCAPTCHA solution entered by the user. The example below (in JSP) shows how this can be done:

```
<%@ page import="net.tanesha.recaptcha.ReCaptchaImpl" %>
<%@ page import="net.tanesha.recaptcha.ReCaptchaResponse" %>

<html>
  <body>
  <%
    String remoteAddr = request.getRemoteAddr();
```

```
        ReCaptchaImpl reCaptcha = new ReCaptchaImpl();
        reCaptcha.setPrivateKey("your_private_key");

        String challenge = request.getParameter("recaptcha_challenge_field");
        String uresponse = request.getParameter("recaptcha_response_field");
        ReCaptchaResponse reCaptchaResponse = reCaptcha.checkAnswer(remoteAddr, challenge, uresponse);

        if (reCaptchaResponse.isValid()) {
          out.print("Answer was entered correctly!");
        } else {
          out.print("Answer is wrong");
        }
      %>
      </body>
    </html>
```

In the code above:

- `remoteAddr` is the user's IP address (which is passed to the reCAPTCHA servers)
- `uresponse` contains the user's answer to the reCAPTCHA challenge.

If you're having trouble, you can also see the following article that explains how to add reCAPTCHA to your Java application: How to reCAPTCHA Your Java Application

Important: DNS Caching

Java has an annoying issue that may cause the connection between your server and reCAPTCHA to be interrupted every few months, and reCAPTCHA will stop working in your site when that happens. Read below to see how to fix this.

By default the Java Virtual Machine (JVM) caches all DNS lookups forever instead of using the time-to-live (TTL) value which is specified in the DNS record of each host. For those of you how do not know it, a DNS lookup is a request sent to a DNS server which converts a readable hostname to an IP address. For example, it converts **www.recaptcha.net** to the IP address **69.12.97.164**. It is of course much more complex than this, and if you want to learn more, wikipedia's entry on DNS is a good starting point.

Although not frequently, reCAPTCHA servers can change IP addresses. Because Java caches DNS lookups forever, this can cause the connection between your server and reCAPTCHA to go down when the reCAPTCHA IP address changes. If this happens, restarting your JVM (e.g., restarting Tomcat) can fix the problem because it causes a new DNS lookup. However, you probably don't want to restart your JVM once every few months whenever your site breaks because the reCAPTCHA servers changed IP address.

To fix this issue for good, you can pass **-Dsun.net.inetaddr.ttl=30** to your app-server (this tells Java to only cache DNS for 30 seconds). In Tomcat for Windows, this can be done by

1. Stop Tomcat
2. Go to tomcat\bin
3. Run Tomcat5w.exe
4. Goto java tab
5. Add java property to java options section: -Dsun.net.inetaddr.ttl=30
6. Exit
7. Start Tomcat

In Tomcat for Linux or MacOS X, you need to run the following command in the command line (and then restart Tomcat):

```
    export JAVA_OPTS="$JAVA_OPTS -Dsun.net.inetaddr.ttl=30"
```

Here is an article explaining more about this issue.

Further Reading

- Customizing Look and Feel
- Tips and Guidelines
- Troubleshooting

Last updated March 20, 2012.