ADF Code Corner

008. How-to use Captcha with ADF Faces and Oracle ADF

twitter.com/adfcodecorner

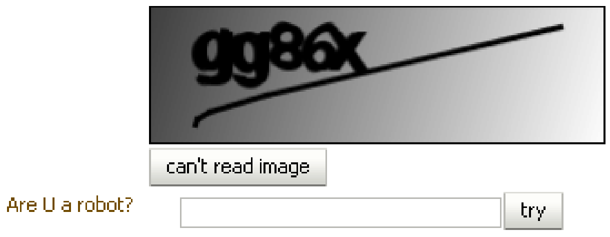
Abstract:

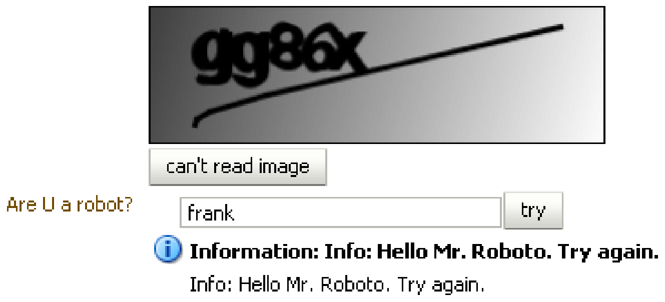
A pictures say more than a thousand words. And it says it in a way that is hard to parse for machines that otherwise would try and spam your system. For sure, you have seen this on the the Internet already and even Google uses it to - at least sometimes - check the seriousness of your query string. Captcha also is very popular when it comes to security. The way it works is that an image is generated and displayed on the screen. The random character string is also made available to the developer, who then can compare the provided user input with what he or she knows

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
17-JULY-2009

Introduction

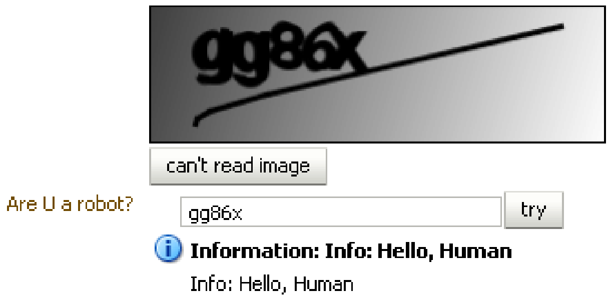
Inspired by a question on OTN, I found a simple captcha solution posted on sourceforge.net that appears to be sufficient for this little ADF Code Corner goodie.





Information: Info: Hello Mr. Roboto. Try again.

Info: Hello Mr. Roboto. Try again.



Information: Info: Hello, Human

Info: Hello, Human

The project is "Simple Captcha" (http://simplecaptcha.sourceforge.net/) and it provides a HTTP servet that you can configure in your ADF Faces application and then reference from an af:image component. SimpleCaptcha is open source software under the BSD License. You can extend the functionality provided by the three simple servlet implementations in your own HttpServlet, following the instructions on the following page:

http://simplecaptcha.sourceforge.net/extending.html

In the Oracle JDeveloper 11g R1 example that you candownload at the end of this article, I let you prove that you are not a robot. A wrong entry that does not match the image text in the captcha will welcome you as Mr. Roboto.

Implementation

The implementation of this sample requires editing of 3 files, web.xml, the ADF Faces page that displays the captcha and a managed bean to initiate the comparison. The web.xml file is edited to contain the servlet reference

```
<servlet>
    <servlet-name>CaptchaServlet</servlet-name>
    <servlet-class>nl.captcha.servlet.SimpleCaptchaServlet</servlet-
class>
    <init-param>
            <param-name>width</param-name>
            <param-value>250</param-value>
    </init-param>
    <init-param>
       <param-name>height</param-name>
       <param-value>75</param-value>
    </init-param>
</servlet>

<servlet-mapping>
    <servlet-name>CaptchaServlet</servlet-name>
    <url-pattern>/captchaservlet</url-pattern>
</servlet-mapping>
```

ADF Faces Page Source

The ADF Page source has nothing remarkable to point out. It uses an af:inputText component for the user answer and anaf:image component to reference the Captcha servlet. An af:message component is used to display the informational message created in the managed bean.

```
  <f:view>
    <af:document id="d1">
      <af:form id="f1">
        <af:panelFormLayout id="pfl1">
          <f:facet name="footer"/>
          <af:image source="/captchaservlet"
                    id="i1" inlineStyle="width:251px; height:76.0px;"/>
          <af:commandButton text="can't read image" id="cb2"
                            partialSubmit="false"/>
          <af:panelLabelAndMessage label="Are U a robot?" id="plam1">
            <af:panelGroupLayout id="pgl1" layout="horizontal"
                                  halign="left">
              <af:inputText id="it1"
                            value="#{requestScope.bestGuess}"/>
              <af:commandButton text="try" id="cb1"
                   actionListener="#{HandleCaptchaBean.verifyAnswer}"
                   partialSubmit="true" immediate="false"/>
            </af:panelGroupLayout>
            <af:message id="m1" messageType="info" for="it1"/>
          </af:panelLabelAndMessage>
        </af:panelFormLayout>
      </af:form>
    </af:document>
  </f:view>
```

Managed Bean Code

```
import java.io.UnsupportedEncodingException;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;
import javax.servlet.http.HttpServletRequest;
import nl.captcha.Captcha;
import oracle.adf.view.rich.context.AdfFacesContext;


public class HandleCaptchBean {
 public HandleCaptchBean() {
}

public void verifyAnswer(ActionEvent actionEvent) {
  FacesContext fctx = FacesContext.getCurrentInstance();
  ExternalContext ectx =  fctx.getExternalContext();

  HttpServletRequest request =
                            (HttpServletRequest) ectx.getRequest();
  Captcha captcha = (Captcha) ectx.getSessionMap().get(Captcha.NAME);
  try {
    request.setCharacterEncoding("UTF-8");
  } catch (UnsupportedEncodingException e) {
    //bad luck - but ignore
    System.out.println("UTF not supported !");
  }
  String answer = (String) ectx.getRequestMap().get("bestGuess");
  if (answer != null && captcha.isCorrect(answer)){
    message("Hello, Human");
  }
  else{
   message("Hello Mr. Roboto. Try again.");
   UIComponent panelLabelAndMessage = ((UIComponent)
   actionEvent.getSource()).getParent().getParent();
   UIComponent panelFormlayout = panelLabelAndMessage.getParent();
   AdfFacesContext.getCurrentInstance().addPartialTarget(
                                                     panelFormlayout);
      }
    }

private void message(String s){
  FacesContext fctx = FacesContext.getCurrentInstance();
  fctx.addMessage("it1",new
      FacesMessage(FacesMessage.SEVERITY_INFO,"Info: "+s,null));
 }
}
```

Download

One of a few samples that do not require a database. The workspace is created with Orace JDeveloper 11*g* R1. There is one area that might need some additional work, which is to refresh the captcha image using PPR. But wait - didn't the title say "with ADF Faces and Oracle ADF"? What is the ADF part in this ? Well, there is none! – Download sample here from the ADF Code Corner website:

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

Update April, 12th 2012

Michael Yehle from Oracle used this sample and found a work around for the refresh issue I faced with this solution. The solution is to expose the Captcha image in an iframe. Here's the work around explained by copying content of Michael's mail:

```
<af:inlineFrame source="/captchaservlet" id="if1"
                clientComponent="true"
                binding="#{viewScope.CaptchBean.iframe}"/>
...
af:commandToolbarButton
      text="Refresh"  id="ctb1"
      actionListener="#{viewScope.CaptchBean.refreshActionListener}"
      immediate="true" partialSubmit="true"/>

and
public void refreshActionListener(ActionEvent actionEvent) {
   AdfFacesContext.getCurrentInstance().addPartialTarget(iframe);
}
```