Upload de fitxers a Canigó 3

A qui va dirigit

Aquest how-to va dirigit a tots aquells que vulguin fer servir el mòdul de pujada d'arxius de Canigó 3.

Versió de Canigó

Els passos descrits en aquest document apliquen a aplicacions Canigó 3 que utilitzin la versió 1.1.0 o posterior del mòdul canigo.web.jsf, i que per tant, utilitzen JSF 2.

Introducció

Donat el creixement de peticions i consultes que arriben relacionades amb el mòdul de pujada de fitxers, des del CS Canigó s'ha realitzat una guia d'ús d'aquest mòdul. En aquesta guia d'ús es seguirà el mateix exemple que figura a la documentació del mòdul, explicant com fer ús del component *t:inputFileUpload* de Tomahawk i afegint informació que pot resultar d'interès.
També es presentarà el component *rich:fileUpload* com a alternativa per a aquells casos en que no es viable fer ús del de tomahawk. I per últim es descriuran les conclusions amb la finalitat de que l'usuari pugui decidir quin fer servir en funció de les seves necessitats.

Upload de fitxers a Canigó 3

Tomahawk

Per a fer ús del component *t:inputFileUpload* cal establir una configuració prèvia:

- Afegir la dependència de Tomahawk al **pom.xml**:

```
<properties>
        ...
        <tomahawk>1.1.14</tomahawk>
</properties>
<dependency>
      <groupId>org.apache.myfaces.tomahawk</groupId>
      <artifactId>tomahawk21</artifactId>
      <version>${tomahawk}</version>
      <exclusions>
        <exclusion>
          <artifactId>commons-validator</artifactId>
          <groupId>commons-validator</groupId>
        </exclusion>
        <exclusion>
          <artifactId>itext</artifactId>
          <groupId>com.lowagie</groupId>
        </exclusion>
        <exclusion>
          <artifactId>jstl</artifactId>
          <groupId>javax.servlet</groupId>
        </exclusion>
        <exclusion>
          <artifactId>commons-logging</artifactId>
          <groupId>commons-logging</groupId>
        </exclusion>
        <exclusion>
          <artifactId>batik-awt-util</artifactId>
          <groupId>org.apache.xmlgraphics</groupId>
```

- Afegir i aplicar el següent filtre al fitxer **web.xml**

```
<filter>
  <filter-name>Extensions Filter</filter-name>
  <filter-class>org.apache.myfaces.webapp.filter.ExtensionsFilter</filter-class>
  <init-param>
    <description>Set the size limit for uploaded files. Format: 10 - 10 bytes, 10k -
10 KB, 10m - 10 MB, 1g - 1 GB</description>
    <param-name>uploadMaxFileSize</param-name>
    <param-value>20m</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>Extensions Filter</filter-name>
  <url-pattern>*.jsf</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>Extensions Filter</filter-name>
  <url-pattern>*.xhtml</url-pattern>
</filter-mapping>
```

Aquesta configuració es pot establir de manera automàtica si es fa mitjançant el plugin de Canigó 3 d'Eclipse, però si es fa d'aquesta manera caldrà tenir en compte el següent:

- S'afegeix la dependència *canigo.support.fileupload* al pom.xml
- S'afegeix el fitxer *fileupload.properties*

Upload de fitxers a Canigó 3

- Caldrà modificar tant la versió de la llibreria de tomahawk (1.1.9 -> 1.1.14) com el nom de l'artefacte de la dependència cap a tomahawk (tomahawk -> tomahawk21). Això es degut al canvi de versió de JSF 1.2 a JSF 2.

Tant la dependència amb *canigo.support.fileupload* com el fitxer *fileupload.properties* s'afegeixen de cara a poder fer servir Struts com a capa de presentació o fer servir la integració amb el mòdul d'antivirus. Si no es pretén fer ús d'aquestes característiques, no seran necessaris i es poden treure.

Una vegada establerta la configuració, es pot implementar el Bean/Controller i la vista que el farà servir:

FileInputTomahawkBean.java

```
package cat.gencat.howtofileupload.bean;

import java.io.IOException;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import org.apache.myfaces.custom.fileupload.UploadedFile;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;
import cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource;

@Component("fileUploadTomahawkBean")
@Scope("singleton")
@Lazy
public class FileUploadTomahawkBean {
  @Autowired
  private I18nResourceBundleMessageSource messageResource;
  private UploadedFile uploadedFile;

  public void submit() {
    try {
      if(getUploadedFile()!=null & getUploadedFile().getBytes()!=null){
        FacesContext.getCurrentInstance().addMessage("uploadForm",
          new FacesMessage(FacesMessage.SEVERITY_INFO,
          messageResource.getMessage("fileUploadSuccess"), null));
      }else{
        FacesContext.getCurrentInstance().addMessage("uploadForm",
          new FacesMessage(FacesMessage.SEVERITY_ERROR,
          messageResource.getMessage("fileUploadError"), null));
      }
    } catch (IOException e) {
      FacesContext.getCurrentInstance().addMessage("uploadForm",
        new FacesMessage(FacesMessage.SEVERITY_ERROR,
        messageResource.getMessage("fileUploadError"), null));
    }
  }

  public UploadedFile getUploadedFile() {
    return uploadedFile;
  }

  public void setUploadedFile(UploadedFile uploadedFile) {
    this.uploadedFile = uploadedFile;
  }
}
```

Cal destacar que la classe `UploadedFile` que s'ha de fer servir és la que pertany al paquet de tomahawk, donat que la mateixa classe existeix també en d'altres paquets.

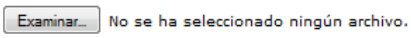
Upload de fitxers a Canigó 3

fileUploadTomahawk.xhtml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:rich="http://richfaces.org/rich"
  xmlns:c="http://java.sun.com/jstl/core"
  xmlns:t="http://myfaces.apache.org/tomahawk">

  <ui:composition template="/views/layouts/template.jsf">
    <ui:define name="body">
      <h:form id="uploadForm" enctype="multipart/form-data">
        <h:panelGrid columns="3">
          <h:outputLabel for="file" value="#{msg.fileUploadSelectFile}" />
          <t:inputFileUpload id="file"
            value="#{fileUploadTomahawkBean.uploadedFile}" required="true" />
          <h:message for="file" style="color: red;" />
          <h:panelGroup/>
          <h:commandButton value="#{msg.canigoSubmit}"
            action="#{fileUploadTomahawkBean.submit}" />
          <h:message for="uploadForm" infoStyle="color: green;"
            errorStyle="color: red;" />
        </h:panelGrid>
      </h:form>
    </ui:define>
  </ui:composition>
</html>
```

L'aspecte visual del renderitzat del component definit per el tag `<t:inputFileUpload>` és el que es pot veure a la següent captura:



Les característiques d'aquest component són les següents:

- Limitacions en canvi d'estils: El component es renderitza com a un <input type="file">, per tant l'aspecte del botó, text dels missatges i idioma d'aquests el determina el navegador, no es poden configurar.
- Limitacions de funcionalitat: El component només deixa escollir un únic arxiu. Per selecció múltiple caldria afegir-ne N components i seleccionar-ne l'arxiu a cada component.
- L'Upload del fitxer es fa efectiu al enviar les dades del formulari.
- El component presenta limitacions amb pàgines que fan servir tecnologia Ajax.

Upload de fitxers a Canigó 3

Richfaces

Per a fer ús del component *<rich:fileUpload>* cal establir una configuració prèvia:

- Afegir els següents paràmetres al fitxer **web.xml:**

```
context-param>
  <param-name>createTempFiles</param-name>
  <param-value>true</param-value>
</context-param>
<context-param>
  <param-name>maxRequestSize</param-name>
  <param-value>1000000</param-value>
</context-param>
```

No cal afegir cap dependència al pom.xml donat que la plantilla de Canigó ja incorpora les dependències amb les llibreries de Richfaces necessàries.

Una vegada establerts aquests paràmetres, es pot implementar el Bean/Controller i la vista que el farà servir:

FileUploadRichfacesBean.java

```
package cat.gencat.howtofileupload.bean;

import java.util.ArrayList;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import org.richfaces.event.FileUploadEvent;
import org.richfaces.model.UploadedFile;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;
import cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource;

@Component("fileUploadRichfacesBean")
@Scope("session")
public class FileUploadRichfacesBean  {

  @Autowired
  private I18nResourceBundleMessageSource messageResource;
  private ArrayList<UploadedFile> files = new ArrayList<UploadedFile>();

  public void listener(FileUploadEvent event) throws Exception {
  UploadedFile item = event.getUploadedFile();
    files.add(item);
  }

  public void submit(){
    //TODO: Fer el tractament dels arxius pujats, ubicats dins de la variable 'files'
    if (files.size()>0){
      FacesContext.getCurrentInstance().addMessage("uploadForm",
        new FacesMessage(FacesMessage.SEVERITY_INFO,
        messageResource.getMessage("fileUploadSuccess"), null));
    }else{
      FacesContext.getCurrentInstance().addMessage("uploadForm",
        new FacesMessage(FacesMessage.SEVERITY_ERROR,
        messageResource.getMessage("fileUploadError "), null));
    }
  }
}
```

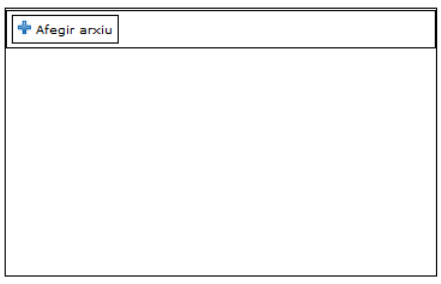
Upload de fitxers a Canigó 3

fileUploadRichfaces.xhtml

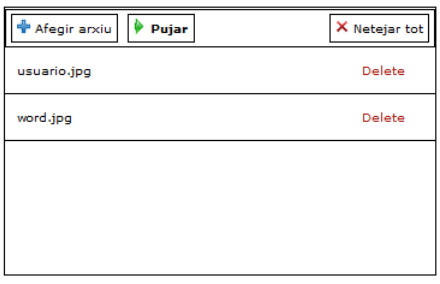
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<html xmlns="http://www.w3.org/1999/xhtml"
       xmlns:ui="http://java.sun.com/jsf/facelets"
       xmlns:f="http://java.sun.com/jsf/core"
       xmlns:h="http://java.sun.com/jsf/html"
       xmlns:rich="http://richfaces.org/rich"
       xmlns:c="http://java.sun.com/jstl/core"
       xmlns:a4j="http://richfaces.org/a4j">

  <ui:composition template="/views/layouts/template.jsf">
    <ui:define name="body">
      <h:form id="uploadForm">
        <h:panelGrid columns="2" columnClasses="top,top">
          <rich:fileUpload id="upload"
            addLabel="#{msg['fileUpload.addLabel']}"
            stopLabel="#{msg['fileUpload.stopLabel']}"
            uploadLabel="#{msg['fileUpload.uploadLabel']}"
            cancelLabel="#{msg['fileUpload.cancelLabel']}"
            clearLabel="#{msg['fileUpload.clearLabel']}"
            clearAllLabel="#{msg['fileUpload.clearAllLabel']}"
            doneLabel="#{msg['fileUpload.doneLabel']}"
            fileUploadListener="#{fileUploadRichfacesBean.listener}" >
            <a4j:ajax event="uploadcomplete" execute="@none" render="info" />
          </rich:fileUpload>
          <h:commandButton value="#{msg.canigoSubmit}"
            action="#{fileUploadRichfacesBean.submit}" />
        </h:panelGrid>
      </h:form>
      <h:message for="uploadForm" infoStyle="color:green;" errorStyle="color:red;" />
    </ui:define>
  </ui:composition>
</html>
```

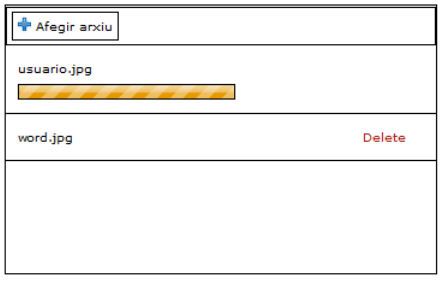
L'aspecte visual (sense aplicar estils normatius) del renderitzat del component definit per el tag *<rich:fileUpload>* és el que es pot veure a les següents captures:



Abans de fer cap selecció d'arxiu



Havent-ne escollit arxius



Al picar sobre el botó de pujar

Havent acabat la pujada

Les característiques d'aquest component són:

- Estils configurables: el component permet sobreescriure els estils per defecte. Per a més informació es pot consultar la tasca JIRA CAN-1816.
- S'han detectat alguns bugs al component (arreglats a darreres versions de Richfaces). Són els següents:
    - L'atribut 'maxFilesQuantity' no funciona
    - L'atribut 'immediateUpload' no funciona

  Aquests bugs s'han pogut resoldre mitjançant el workaround que figura a la tasca CAN-1825 (redefinir el comportament de la funció __updateButtons que fa servir el component)
- L'upload es fa efectiu al picar sobre el botó de Upload que renderitza el component, no al enviar el formulari que el conté.
- Permet pujar diversos arxius alhora, però la selecció d'aquests es fa de manera individual, mitjançant finestra modal d'escollir arxiu amb selecció única.

Conclusions

A mode de conclusió a continuació es mostra un llistat de les principals característiques de cada component. Amb aquesta informació l'usuari pot decidir quin dels dos components s'ajusta més a les necessitats de la seva aplicació:

Tomahawk (Mòdul d'Upload de Fitxers)	Richfaces
- Simplicitat - Enviament en submit de formulari (no Ajax) - No sel·lecció múltiple - No filtratge per extensió de fitxer - Limitació tamany fitxer configurable - No cal definir estils	- Més complex que el de Tomahawk - Enviament en sel·lecció de fitxer (Ajax) - Sel·lecció múltiple (limitació de número de fitxers configurable) - Filtratge per extensió def fitxers - Limitació tamany de fitxer configurable - Necessitat d'adaptació a estils normatius