C:\DATEN\Projects\Studium\openpatternrepository\30_sources\PatternRepository\web-interface\src\main\java\nl\rug\search\opr\backingbean\UploadHelper.java

```java
package nl.rug.search.opr.backingbean;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Map;
import javax.annotation.PostConstruct;
import javax.ejb.EJB;
import javax.faces.context.FacesContext;
import nl.rug.search.opr.entities.pattern.File;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.component.UIComponent;
import nl.rug.search.opr.file.FileException;
import nl.rug.search.opr.entities.pattern.License;
import nl.rug.search.opr.entities.pattern.PatternVersion;
import nl.rug.search.opr.file.FileLocal;
import org.icefaces.component.fileentry.FileEntry;
import org.icefaces.component.fileentry.FileEntryEvent;
import org.icefaces.component.fileentry.FileEntryResults;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 *
 * @author Martin Verspai <martin@verspai.de>
 * @version 2.0
 * @date 26.10.2009
 */
@ManagedBean(name = "uploadHelper")
@ViewScoped
public class UploadHelper {


    protected PatternVersion version;
    private License license;
    private FileEntryResults.FileInfo fileInfo;
    private String name;
    private UIComponent uploadComponent;
    @EJB
    private FileLocal fileBean;
    private Logger logger = LoggerFactory.getLogger(UploadHelper.class);


    public UploadHelper() {
    }

    @PostConstruct
    public void init() {
        FacesContext ctx = FacesContext.getCurrentInstance();
        String beanName = ctx.getExternalContext().getRequestParameterMap().get("patternVersion");

        version = findBean(beanName, PatternVersion.class);
    }

    public static <T> T findBean(String beanName, Class<T> beanClass) {
        FacesContext context = FacesContext.getCurrentInstance();
        return beanClass.cast(context.getApplication().evaluateExpressionGet(context, "#{" + beanName + "}", beanClass));
    }

    public PatternVersion getVersion() {
        return version;
    }

    public void setVersion(PatternVersion version) {
```

```
66      this.version = version;
67  }
68
69  public void listener(FileEntryEvent event) {
70      FileEntry fileEntry = (FileEntry) event.getSource();
71      FileEntryResults results = fileEntry.getResults();
72
73      for (FileEntryResults.FileInfo fi : results.getFiles()) {
74          FacesContext context = FacesContext.getCurrentInstance();
75          if (fi.isSaved()) {
76              fileInfo = fi;
77              name = fi.getFileName();
78              saveFile(fileInfo, name);
79          }
80      }
81
82  }
83
84  private void saveFile(FileEntryResults.FileInfo fileInfo, String name) {
85
86
87      try {
88
89          File file = null;
90          if (fileInfo.getStatus().isSuccess()) {
91
92              String fileName = fileInfo.getFileName();
93              file = fileBean.add(license, fileName, new FileInputStream(fileInfo.getFile()));
94          }
95          version.getFiles().add(file);
96
97      } catch (FileNotFoundException ex) {
98          System.out.println("File not found");
99      } catch (IOException ex) {
100          System.out.println("IO exception");
101      } catch (FileException ex) {
102      } finally {
103          java.io.File physicalFile = fileInfo.getFile();
104          if (physicalFile.exists()) {
105              physicalFile.delete();
106          }
107          fileInfo = null;
108          name = "";
109      }
110
111  }
112
113  public String getName() {
114      return name;
115  }
116
117  public void setName(String name) {
118      this.name = name;
119  }
120
121  public License getLicense() {
122      return license;
123  }
124
125  public void setLicense(License license) {
126      this.license = license;
127  }
128
129  public UIComponent getUploadComponent() {
130      return uploadComponent;
131  }
132
133  public void setUploadComponent(UIComponent uploadComponent) {
```

```
134         this.uploadComponent = uploadComponent;
135     }
136 
137     public String getSupportedFileTypes() {
138         String out = "";
139         for (String fileType : fileBean.getSupportedMimeTypes()) {
140 
141             String[] split = fileType.split("/");
142 
143             if (split.length == 2) {
144                 out = out.concat("." + split[1] + ", ");
145             }
146         }
147         out = out.substring(0, out.length() - 2);
148 
149         return out;
150     }
151 
152     public int getMaximumFileSize() {
153         return fileBean.getMaximumFileSizeMb();
154     }
155 }
156 
157 
```