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

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;
import javax.servlet.http.HttpServletRequest;
import nl.rug.search.opr.AbstractFormBean;
import nl.rug.search.opr.component.ConsequenceWrapper;
import nl.rug.search.opr.component.ForceWrapper;
import nl.rug.search.opr.component.TextBlockWrapper;
import nl.rug.search.opr.entities.pattern.Consequence;
import nl.rug.search.opr.entities.pattern.Force;
import nl.rug.search.opr.entities.pattern.Pattern;
import nl.rug.search.opr.pattern.PatternLocal;
import nl.rug.search.opr.entities.pattern.PatternVersion;
import nl.rug.search.opr.entities.pattern.TextBlock;
import nl.rug.search.opr.entities.template.Component;
import nl.rug.search.opr.entities.pattern.File;

/**
 *
 * @author Martin Verspai <martin@verspai.de>
 * @date 07.12.2009
 */
@ManagedBean(name="editVersionCtrl")
@SessionScoped
public class EditVersionController extends AbstractFormBean {

    @EJB
    private PatternLocal pl;

    private static final String PATTERN_ID = "patternId";
    private static final String VERSION_ID = "versionId";

    private static final String successMsg   = "Pattern has been edited!";
    private static final String failMsg      = "Pattern has not been edited!";

    private Pattern pattern;
    private PatternVersion patternVersion;
    private Map<String, TextBlockWrapper> blocks;
    private Collection<ForceWrapper> forces;
    private Collection<ConsequenceWrapper> consequences;

    public EditVersionController() {
        init();
    }

    @Override
    public String getFormId() {
        return "editDescription";
    }

    @Override
    public String successMessage() {
        return EditVersionController.successMsg;
    }

    @Override
```

```
public String failMessage() {
    return EditVersionController.failMsg;
}

public void reset(ActionEvent e) {
    reset();
}

@Override
public void reset() {
    initData();
}

private void init() {
    this.pattern         = null;
    this.patternVersion = null;
    this.blocks          = new HashMap<String, TextBlockWrapper>();
    this.forces          = new ArrayList<ForceWrapper>();
    this.consequences    = new ArrayList<ConsequenceWrapper>();
}


@Override
public void execute() {
    List<TextBlock> tmpBlocks             = new ArrayList<TextBlock>();
    List<Force> patternForces             = new ArrayList<Force>();
    List<Consequence> patternConsequences = new ArrayList<Consequence>();

    for (TextBlockWrapper tbw : blocks.values()) {
        TextBlock tb = tbw.getTextBlock();
            tb.setId(null);
        tmpBlocks.add(tb);
    }

     for(ForceWrapper fw : this.forces) {
         Force f = fw.getForce();
             f.setId(null);
         patternForces.add(f);
     }

     for(ConsequenceWrapper cw : this.consequences) {
         Consequence c = cw.getConsequence();
             c.setId(null);
         patternConsequences.add(c);
     }

     PatternVersion newVersion = new PatternVersion();
         newVersion.setAuthor(this.patternVersion.getAuthor());
         newVersion.setBlocks(tmpBlocks);
         newVersion.setConsequences(patternConsequences);
         newVersion.setForces(patternForces);
         newVersion.setLicense(this.patternVersion.getLicense());
         newVersion.setSource(this.patternVersion.getSource());
         newVersion.setTemplate(this.patternVersion.getTemplate());
         newVersion.setFiles((List<File>)this.patternVersion.getFiles());

     this.pattern.setCurrentVersion(newVersion);

     pl.editVersion(this.pattern);

     Pattern p = pl.getById(this.pattern.getId());

     this.pattern = p;
     this.patternVersion = p.getCurrentVersion();

     initData();
 }

```

```java
public Pattern getPattern() {
    HttpServletRequest request = (HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext().getRequest();

    String patternIdStr = request.getParameter(PATTERN_ID);
    String versionIdStr = request.getParameter(VERSION_ID);

    if( patternIdStr != null && !patternIdStr.equals("") ) {
        Pattern tmpPattern;
        PatternVersion tmpVersion;

        try {
            Long tmpId = Long.parseLong(patternIdStr);

            if( (tmpPattern = pl.getById(tmpId)) != null) {
                tmpVersion = tmpPattern.getCurrentVersion();

                if( versionIdStr != null && !versionIdStr.equals("") ) {
                    long versionId = Integer.parseInt(versionIdStr);

                    for(PatternVersion pv : tmpPattern.getVersions()) {
                        if( pv.getId().equals(versionId) ) {
                            tmpVersion = pv;
                        }
                    }
                }

                if(this.pattern == null || !this.pattern.equals(tmpPattern)) {
                    this.pattern = tmpPattern;
                    this.patternVersion = tmpVersion;

                    initData();
                }

                if(this.patternVersion == null || !this.patternVersion.equals(tmpVersion)) {
                    this.patternVersion = tmpVersion;

                    initData();
                }
            }
        } catch (NumberFormatException nfe) { /* TODO: Do something? */ }
    }

    return this.pattern;
}

private void initData() {
    List<File> uploads = new ArrayList<File>();
        uploads.addAll(this.patternVersion.getFiles());

    this.blocks         = new HashMap<String, TextBlockWrapper>();
    this.forces         = new ArrayList<ForceWrapper>();
    this.consequences   = new ArrayList<ConsequenceWrapper>();

    for (Component c : this.patternVersion.getTemplate().getTextComponents()) {
        if (!blocks.containsKey(c.getIdentifier())) {
            TextBlock block = new TextBlock();
            block.setComponent(c);
            block.setText("");
            blocks.put(c.getIdentifier(), new TextBlockWrapper(block));
        }
    }

    for (TextBlock tb : this.patternVersion.getBlocks()) {
        blocks.put(tb.getComponent().getIdentifier(), new TextBlockWrapper(tb));
    }

    for (Force f : this.patternVersion.getForces() ) {
        this.forces.add(new ForceWrapper(f));
    }

    for (Consequence c : this.patternVersion.getConsequences() ) {
        this.consequences.add(new ConsequenceWrapper(c));
    }
}

public PatternVersion getVersion() {
    return this.patternVersion;
}

public Map<String, TextBlockWrapper> getBlocks() {
    return this.blocks;
}

public Collection<ForceWrapper> getForces() {
    return forces;
}

public void setForces(Collection<ForceWrapper> forces) {
    this.forces = forces;
}

 public void addForce(ActionEvent e) {
    ForceWrapper fw = new ForceWrapper(new Force());
    fw.setEditMode(true);
    forces.add(fw);
}

public void removeForce(ActionEvent e) {
    ForceWrapper f = (ForceWrapper) e.getComponent().getAttributes().get("force");
    forces.remove(f);
}

public Collection<ConsequenceWrapper> getConsequences() {
    return this.consequences;
}

public void setConsequences(Collection<ConsequenceWrapper> consequences) {
    this.consequences = consequences;
}

public void addConsequence(ActionEvent e) {
    ConsequenceWrapper cw = new ConsequenceWrapper(new Consequence());
    cw.setEditMode(true);
    this.consequences.add(cw);
}

public void removeConsequence(ActionEvent e) {
    ConsequenceWrapper c = (ConsequenceWrapper) e.getComponent().getAttributes().get("consequence");
    this.consequences.remove(c);
}

public void removeFile(ActionEvent e) {
    File f = (File) e.getComponent().getAttributes().get("file");
    if (f != null) {
       patternVersion.getFiles().remove(f);
    }
}

}


```