ADF – Upload & Download файлов

Демонстрационный проект для версии **JDeveloper 12c** можно скачать по следующей ссылке:
http://buhgalter-online.kz/files/j2ee/adf/UploadDemo.rar

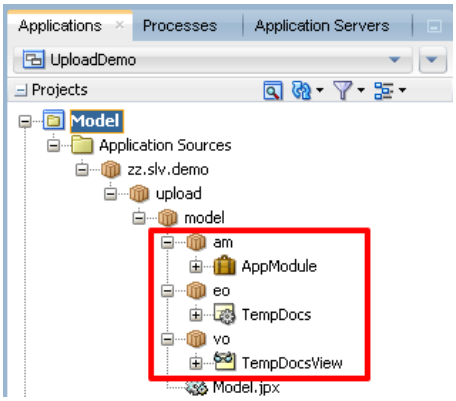
Подготовительные процедуры

Для демонстрации создадим таблицу и сиквенс в БД:

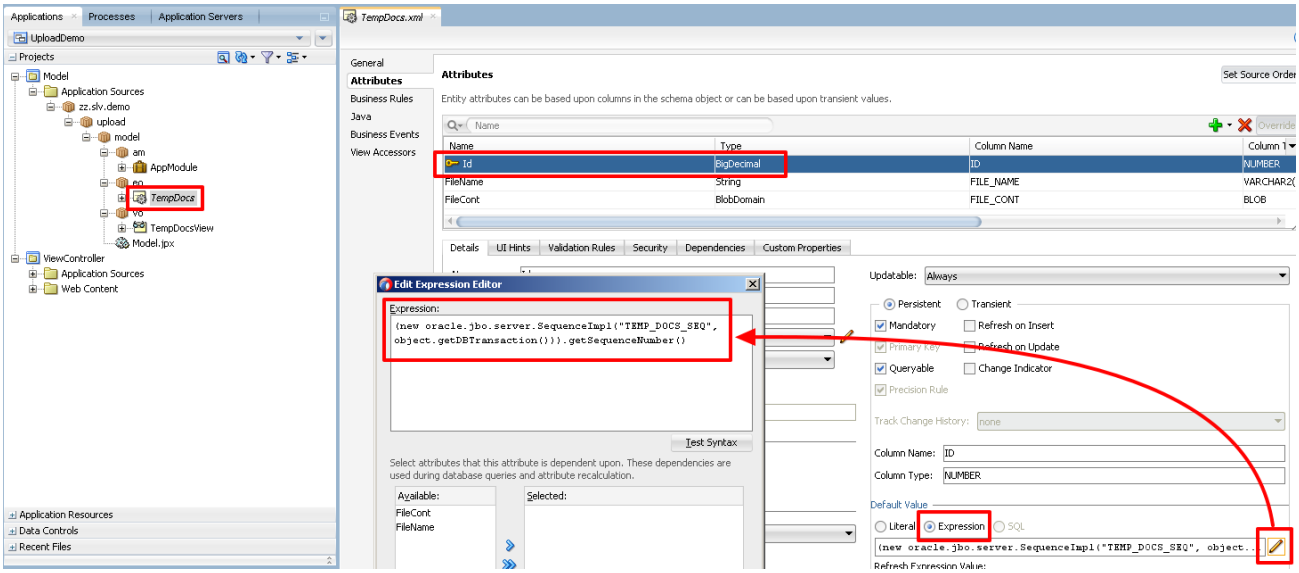
```
DROP TABLE TEMP_DOCS;
CREATE TABLE TEMP_DOCS(
  ID          NUMERIC NOT NULL PRIMARY KEY,
  FILE_NAME   VARCHAR2(500 CHAR),     -- имя файла
  FILE_CONT   BLOB                    -- содержимое файла
);

CREATE SEQUENCE TEMP_DOCS_SEQ;
```

При помощи мастера создадим **Entity** и **View** объекты для таблицы **TEMP_DOCS**:

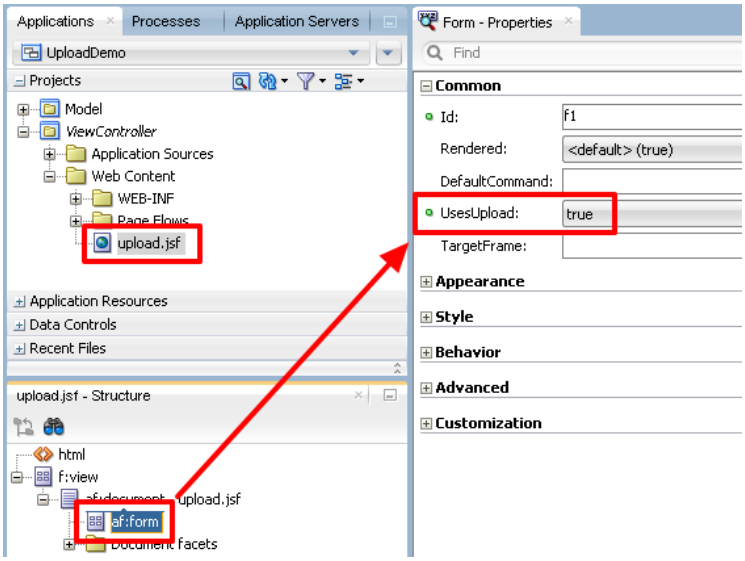


Пропишем выражение для получения нового значения сиквенса:

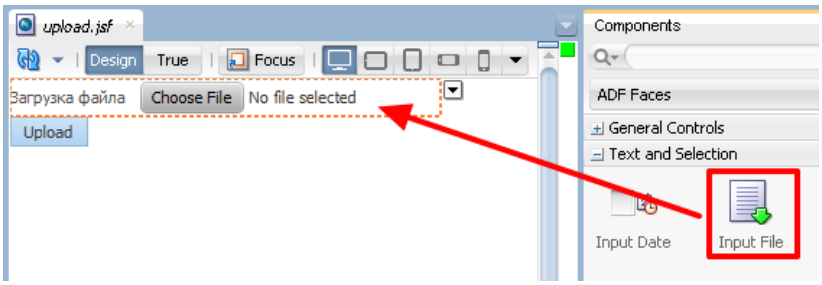


Создание страницы для работы с файлами

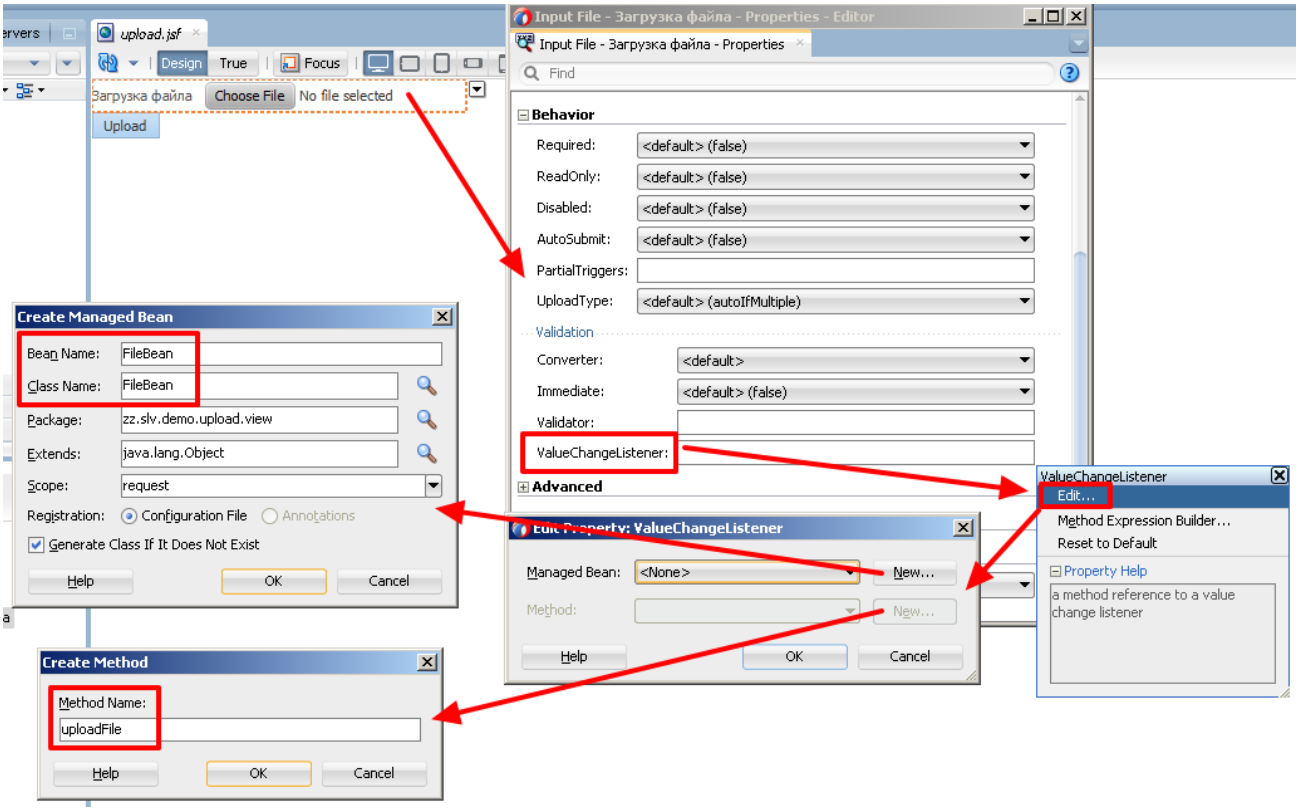
Создадим страницу **upload.jsf** и укажем у элемента **form** **UserUpload = true**:



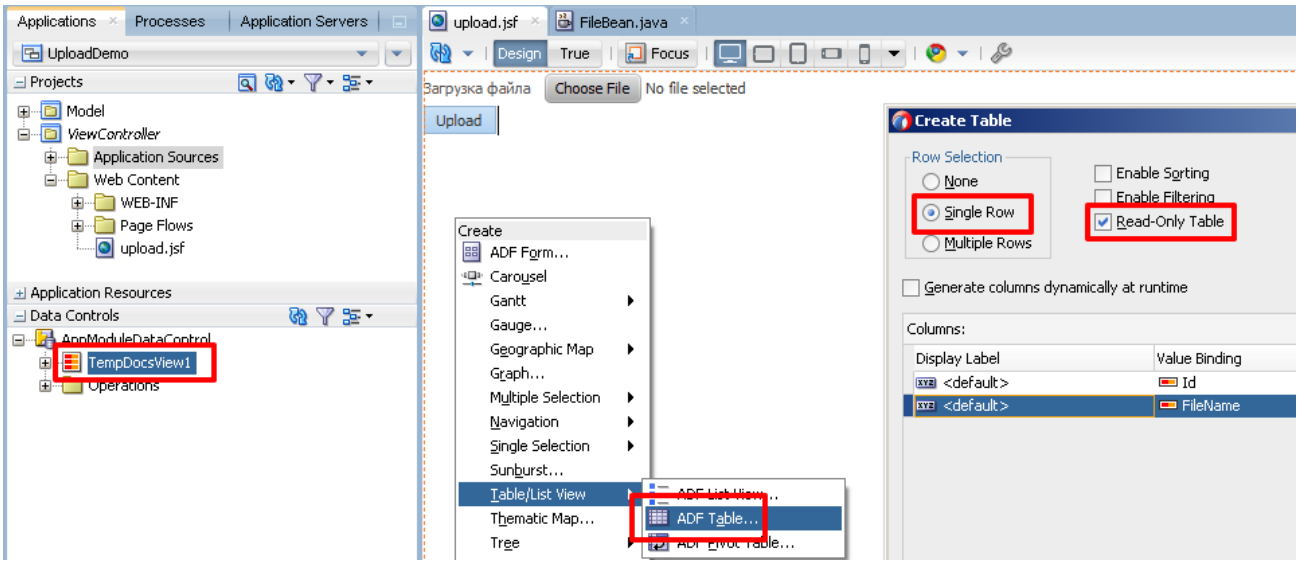
Выложим на страницу компонент **Input File** и кнопку, которой будет осуществляться Submit:



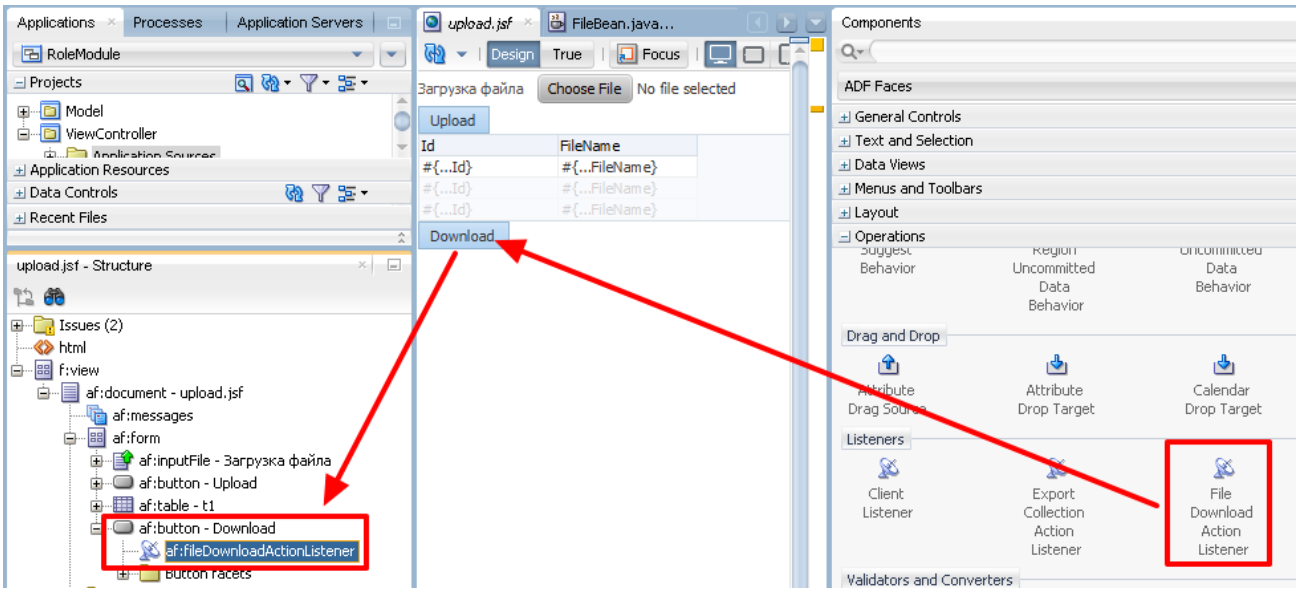
Создадим метод для **ValueChangeListener**:



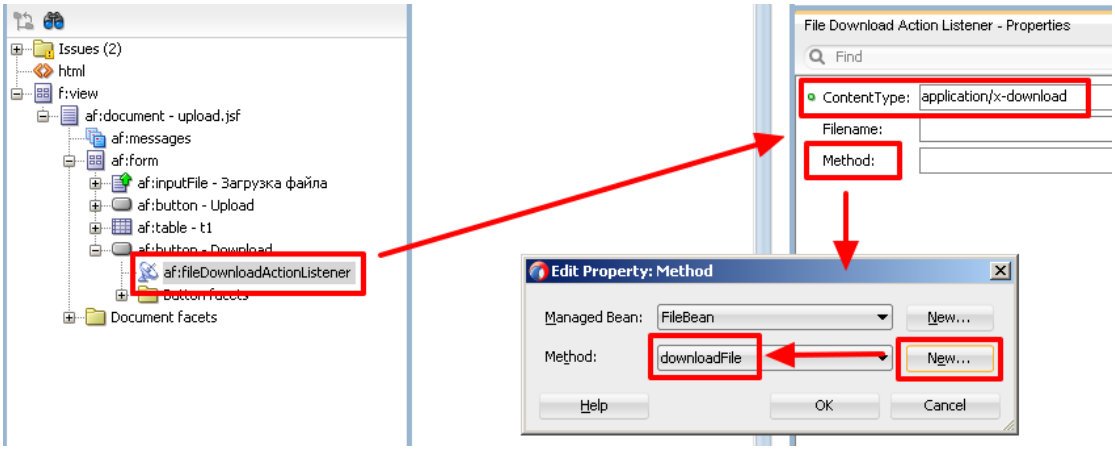
Так же здесь создадим таблицу, которая будет отражать загруженные файлы:



И создадим кнопку **Download**, на которую выложим компонент **File Download Action Listener**:



Создадим для **File Download Action Listener** метод для реализации Download:



Реализация бина реализующего функции Upload и Download

```
package zz.slv.demo.upload.view;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.URLEncoder;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import javax.faces.context.FacesContext;
import javax.faces.event.ValueChangeEvent;
import javax.servlet.http.HttpServletResponse;
import oracle.adf.model.binding.DCIteratorBinding;
import oracle.jbo.Row;
import oracle.jbo.domain.BlobDomain;
import oracle.jbo.server.ViewObjectImpl;
import org.apache.myfaces.trinidad.model.UploadedFile;

public class FileBean {
    private static final String FILE_ITERATOR_NAME = "TempDocsView1Iterator";

    private DCIteratorBinding fileIterator;
    private ViewObjectImpl fileVO;

    public FileBean() {
        // получаем ссылку на итератор
        fileIterator = Util.getDCIterator(FILE_ITERATOR_NAME);
        // получаем ссылку на View Object
        fileVO = (ViewObjectImpl) fileIterator.getViewObject();

    }

    /**
     * Обработка загруженого файла - Upload
     * @param valueChangeEvent
     */
    public void uploadFile(ValueChangeEvent valueChangeEvent) {
        System.out.println("uploadFile - START");

        if (valueChangeEvent.getNewValue() != null) {
            saveFileToBlob((UploadedFile) valueChangeEvent.getNewValue());
        }

        System.out.println("uploadFile - FINISH");
    }

    /**
     * сохранение файла в BLOB
     * @param fileInfo
     */
    private void saveFileToBlob(UploadedFile fileInfo) {
        System.out.println("saveFileToBlob - START");

        try {
            // создаем новую строку
            Row row = fileVO.createRow();

            // имя файла и его содержимое
            row.setAttribute("FileName", fileInfo.getFilename());
            row.setAttribute("FileCont", createBlobDomain(fileInfo));

            // добавляем строку во VO
            fileVO.insertRow(row);

            fileVO.getApplicationModule().getTransaction().commit();
        } catch (Exception e) {
            e.printStackTrace();
```

```
            fileVO.getApplicationModule().getTransaction().rollback();
        }

        System.out.println("saveFileToBlob - FINISH");
    }

    /**
     * Создает BlobDomain по данным UploadedFile
     * @param file
     * @return
     */
    private BlobDomain createBlobDomain(UploadedFile file) {
        System.out.println("createBlobDomain - START");
        InputStream in = null;
        BlobDomain blobDomain = null;
        OutputStream out = null;
        try {
            in = file.getInputStream();
            blobDomain = new BlobDomain();
            out = blobDomain.getBinaryOutputStream();
            writeFromInputToOutput(in, out);
        } catch (SQLException e) {
            e.fillInStackTrace();
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        System.out.println("createBlobDomain - FINISH");
        return blobDomain;
    }

    private static final int BUFFER_SIZE = 1024 * 4;
    private static final int EOF_MARK = -1;
    /**
     * Полное копирование данных из одного потока в другой
     * @param source
     * @param dest
     * @return
     */
    public static int writeFromInputToOutput(InputStream source, OutputStream dest) {
        System.out.println("writeFromInputToOutput - START");
        byte[] buffer = new byte[BUFFER_SIZE];
        int bytesRead = EOF_MARK;
        int count = 0;
        try {
            while ((bytesRead = source.read(buffer)) != EOF_MARK) {
                dest.write(buffer, 0, bytesRead);
                count += bytesRead;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("writeFromInputToOutput - FINISH");
        return count;
    }

    /**
     * Download файла
     * @param facesContext
     * @param outputStream
     */
    public void downloadFile(FacesContext facesContext, OutputStream outputStream) {
        System.out.println("downloadFile - START");

        try {
            Row row = fileVO.getCurrentRow();
            System.out.println(row.getAttribute("Id"));

            // получаем имя файла
            String fileName = (String) row.getAttribute("FileName");
```

```
            // кодируем имя файла для корректного считывания браузером
            String encodedFileName = URLEncoder.encode(fileName, "UTF-8").replace("+", "%20");
            System.out.println("File Name = " + fileName + "; Encoded File Name = " + encodedFileName);

            HttpServletResponse response =
                (HttpServletResponse) facesContext.getCurrentInstance().getExternalContext().getResponse();

            // следущее значение задано в атрибуте contentType fileDownloadActionListener
            //response.setContentType("application/x-download");

            // задаем имя файла получаемое браузером
            response.setHeader("Content-Disposition", "attachment;filename=\"" + encodedFileName + "\"");

            // BLOB - содержимое файла
            BlobDomain blob = (BlobDomain) row.getAttribute("FileCont");

            // копируем содержимое BLOB в отдаваемый поток
            InputStream in = blob.getInputStream();

            //OutputStream out = response.getOutputStream();
            OutputStream out = outputStream;
            writeFromInputToOutput(in, out);

            in.close();

            out.flush();
            out.close();

            facesContext.responseComplete();

        } catch (Exception e) {
            e.printStackTrace();
        }

        System.out.println("downloadFile - FINISH");
    }
}
```

Мультизагрузка

Для реализации загрузки файлов группой, необходимо у компонента **Input File** выставить свойство **maximumFiles = максимальное кол-во загружаемых файлов**, и переписать метод **uploadFile** примерно следующим образом:

```
    public void uploadFile(ValueChangeEvent valueChangeEvent) {
        System.out.println("uploadFile - START");

        if (valueChangeEvent.getNewValue() != null) {

            List<String> alreadyProcessedFiles = new ArrayList<String>();

            ArrayList<UploadedFile> filesArray = (ArrayList) valueChangeEvent.getNewValue();

            for (UploadedFile fileInfo : filesArray) {
                String fileName = fileInfo.getFilename();

                // в 12 версии ADF баг - при мультизагрузке дублируются файлы
                if (!alreadyProcessedFiles.contains(fileName)) {
                    alreadyProcessedFiles.add(fileName);
                    System.out.println(fileName);
                    saveFileToBlob(fileInfo);
                }
            }
        }

        System.out.println("uploadFile - FINISH");
    }
```

Максимальный размер закачиваемых файлов

Задается в байтах, в параметре «**org.apache.myfaces.trinidad.UPLOAD_MAX_DISK_SPACE**» находящегося в «**web.xml**» приложения: