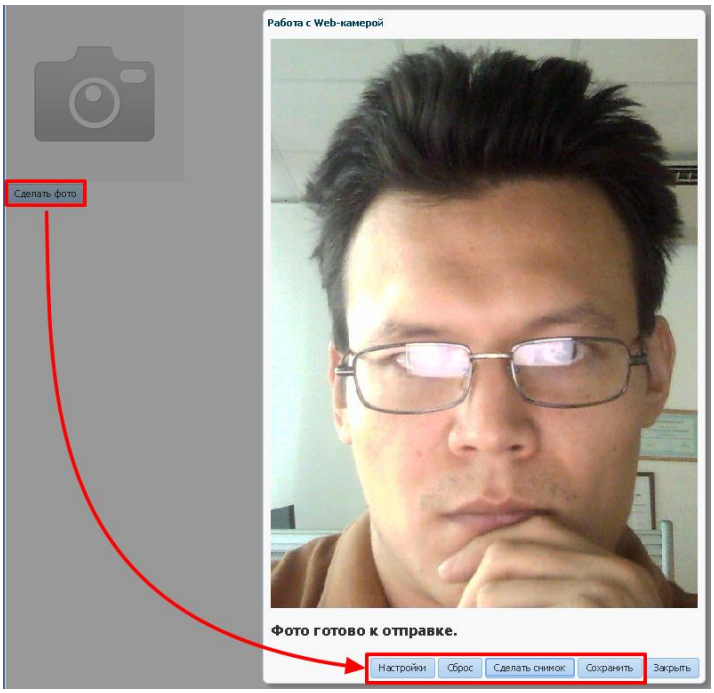
ADF – Работаем с Web-камерой через Flash плагин jpegcam

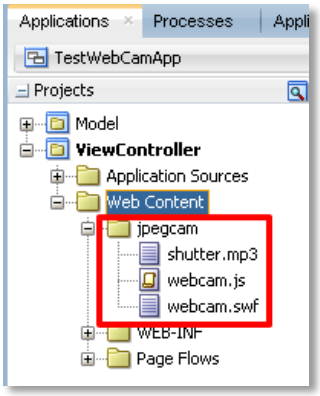


Демонстрационный проект для версии JDeveloper 12c можно скачать по следующей ссылке:
http://buhgalter-online.kz/files/j2ee/adf/TestWebCamApp.rar

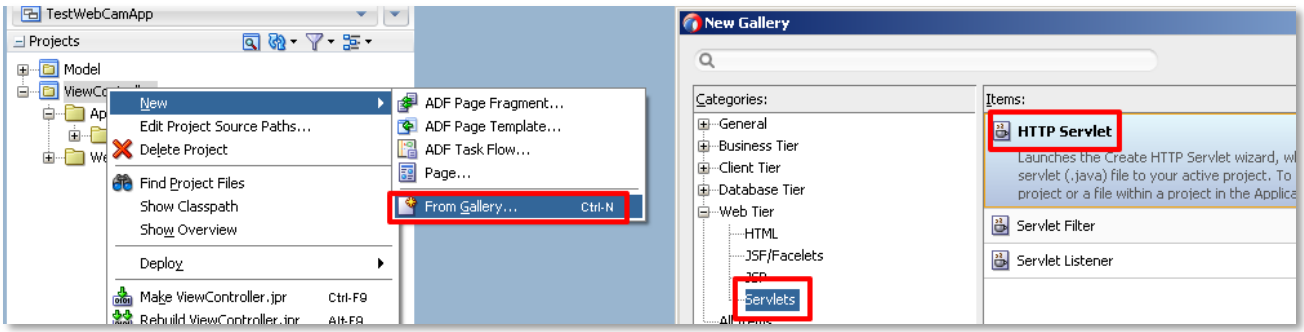
Скачать Flash-плагин **jpegcam** и ознакомиться с его описанием можно по ссылке:
https://code.google.com/p/jpegcam/

Описание демонстрационного проекта

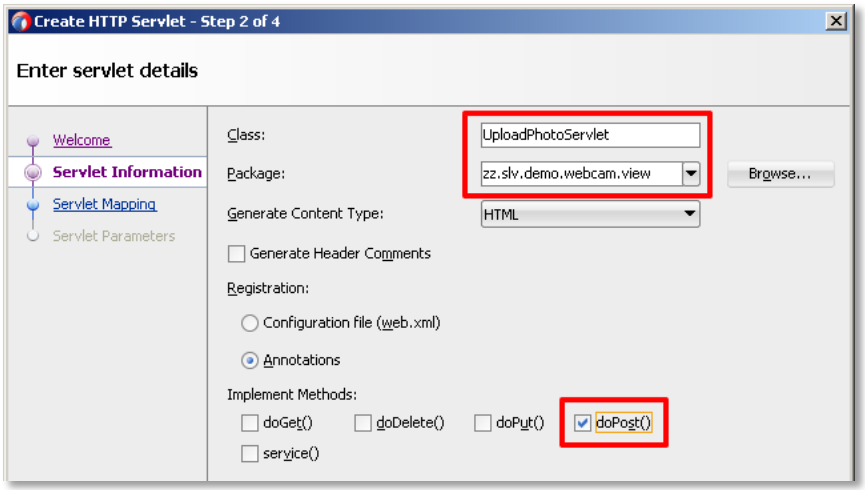
Добавим следующие файлы в каталог **public_html** нашего проекта:

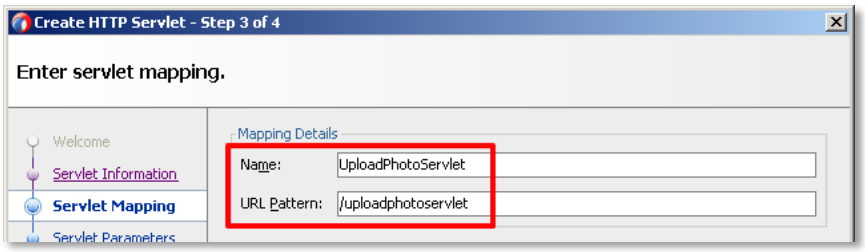


Создадим 2 вспомогательных сервлета, **первый для отправки фото** на сервер, **второй для получения фото**.
Сервлеты можно создать при помощи мастера:



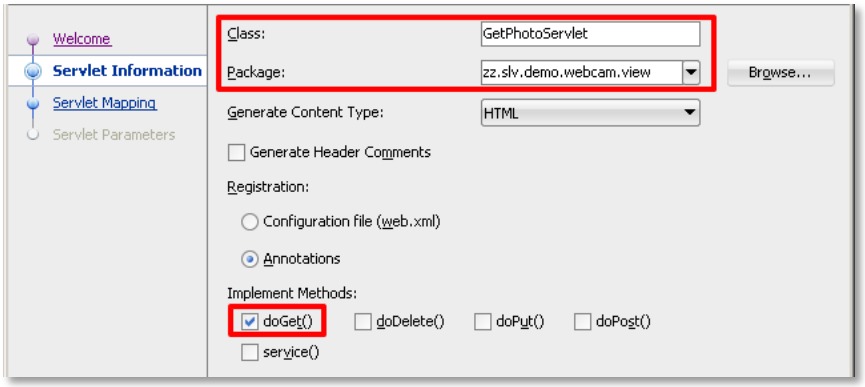
Сервлет **UploadPhotoServlet**:

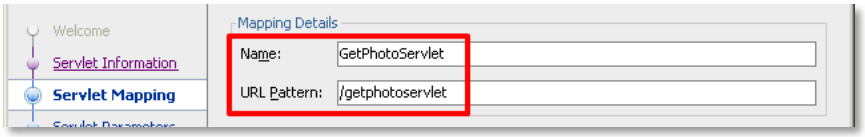





```
@WebServlet(name = "UploadPhotoServlet", urlPatterns = { "/uploadphotoservlet" })
public class UploadPhotoServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=UTF-8";
    private static final String PHOTO_PATH = "c:/temp/photo.jpg";

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
        System.out.println("UploadPhotoServlet - START");
        response.setContentType(CONTENT_TYPE);

        PrintWriter out = response.getWriter();

        BufferedInputStream in = null;
        FileOutputStream fout = null;

        try {
            in = new BufferedInputStream(request.getInputStream());
            fout = new FileOutputStream(PHOTO_PATH);
            Util.writeFromInputToOutput(in, fout);
            out.print("ok");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (in != null) {
                in.close();
            }
            if (fout != null) {
                fout.close();
            }
        }

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

Сервлет **GetPhotoServlet**:






```
@WebServlet(name = "GetPhotoServlet", urlPatterns = { "/getphotoservlet" })
public class GetPhotoServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "image/jpg";
    private static final String PHOTO_PATH = "c:/temp/photo.jpg";

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
                      throws ServletException, IOException {
        System.out.println("GetPhotoServlet - START");

        response.setContentType(CONTENT_TYPE);

        OutputStream out = response.getOutputStream();
        FileInputStream in = null;

        try {

            File f = new File(PHOTO_PATH);

            if (f.exists()) {
                in = new FileInputStream(f);
            } else {
                String blankImagePath = getServletConfig().getServletContext().getRealPath("") +
                                        "/images/no_photo.jpg";
                in = new FileInputStream(blankImagePath);
            }

            // копируем содержимое файла в отдаваемый поток
            Util.writeFromInputToOutput(in, out);

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (in != null) {
                in.close();
            }
        }

        out.close();

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

Вспомогательный класс Util:

```
public class Util {
    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) {
        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();
        }
        return count;
    }
}
```

Бин для страницы:

```
public class WebCamBean {
    private RichImage imagePhoto;
    private RichPopup popupWebCam;

    public WebCamBean() {
    }

    /**
     * Возвращает текущее время в миллисекундах
     * @return
     */
    public long getCurrentTimeMillis() {
        return System.currentTimeMillis();
    }

    /**
     * Скрывает диалог работы с Web-камерой
     * @param actionEvent
     */
    public void closeWebCamDlg(ActionEvent actionEvent) {
        popupWebCam.hide();
        AdfFacesContext.getCurrentInstance().addPartialTarget(imagePhoto);
    }

    public void setImagePhoto(RichImage imagePhoto) {
        this.imagePhoto = imagePhoto;
    }

    public RichImage getImagePhoto() {
        return imagePhoto;
    }

    public void setPopupWebCam(RichPopup popupWebCam) {
        this.popupWebCam = popupWebCam;
    }

    public RichPopup getPopupWebCam() {
        return popupWebCam;
    }
}
```

Тестовая JSF-страница:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE html>
<f:view xmlns:f="http://java.sun.com/jsf/core" xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
    <af:document title="testWebCamPage.jsf" id="d1">
        <af:form id="f1">
            <af:panelStretchLayout id="psl1" startWidth="0" endWidth="0" topHeight="0" bottomHeight="0"
                                   dimensionsFrom="parent">
                <af:resource type="javascript" source="../jpegcam/webcam.js"/>
                <af:resource type="javascript">
                  // подготовка Flash-плагина
                  function webcam_prepare() {
                      webcam.set_api_url('../uploadphotoservlet');
                      // JPEG quality (1 - 100)
                      webcam.set_quality(90);
                      // play shutter click sound
                      webcam.set_shutter_sound(true, '../jpegcam/shutter.mp3');
                      webcam.set_hook('onComplete', 'webcam_completion_handler');

                      // изменяем ссылку на ролик
                      webcam.set_swf_url('../jpegcam/webcam.swf');
                      // вывод Flash-плагина
                      document.getElementById('webcam_flash').innerHTML =
                                                              webcam.get_html(480, 640, 480, 640);

                      webcam.photo_is_ready = false;
                      document.getElementById('upload_results').innerHTML = 'Камера готова.';
                  }
                  // настройка
                  function webcam_config() {
                      webcam.configure();
                  }
                  // сброс камеры для нового фото
                  function webcam_reset() {
                      // reset camera for another shot
                      webcam.reset();
                      webcam.photo_is_ready = false;
                      document.getElementById('upload_results').innerHTML = 'Камера готова.';
                  }
                  // сделать снимок
                  function webcam_capture() {
                      if (!webcam.photo_is_ready) {
                          webcam.freeze();
                          webcam.photo_is_ready = true;
                          document.getElementById('upload_results').innerHTML = 'Фото готово к отправке.';
                      }
                  }
                  // загрузка фото на сервер
                  function webcam_upload() {
                      if (webcam.photo_is_ready) {
                          // take snapshot and upload to server
                          document.getElementById('upload_results').innerHTML = 'Загрузка...';
                          webcam.upload();
                      }
                      else {
                          alert('Сначала необходимо сделать снимок!');
                      }
                  }
                  // срабатывает после окончания загрузки
                  function webcam_completion_handler(msg) {
                      if (msg === 'ok') {
                          webcam_reset();
                      }
                      else {
                          alert('Error');
                      }
                  }
                </af:resource>
```

```
                <f:facet name="bottom">
                    <af:popup childCreation="deferred" id="pWebCam"
                              binding="#{backingBeanScope.WebCamBean.popupWebCam}">
                        <af:clientListener method="webcam_prepare" type="popupOpened"/>
                        <af:dialog id="d2" type="none" closeIconVisible="false" title="Работа с Web-камерой">
                            <div id="webcam_flash"></div>
                            <h2 id="upload_results"></h2>
                            <f:facet name="buttonBar">
                                <af:button text="Настройки" id="b10">
                                    <af:clientListener method="webcam_config" type="click"/>
                                </af:button>
                                <af:button text="Сброс" id="b9">
                                    <af:clientListener method="webcam_reset" type="click"/>
                                </af:button>
                                <af:button text="Сделать снимок" id="b12">
                                    <af:clientListener method="webcam_capture" type="click"/>
                                </af:button>
                                <af:button text="Сохранить" id="b11">
                                    <af:clientListener method="webcam_upload" type="click"/>
                                </af:button>
                                <af:button id="b3" text="Закрыть"
                                     actionListener="#{backingBeanScope.WebCamBean.closeWebCamDlg}"/>
                            </f:facet>
                        </af:dialog>
                    </af:popup>
                </f:facet>
                <f:facet name="center">
                    <af:panelGroupLayout id="pgl9" layout="vertical">
                        <af:image id="i1"
                             source="/getphotoservlet?x=#{backingBeanScope.WebCamBean.currentTimeMillis}"
                             inlineStyle="height:200px;"
                             binding="#{backingBeanScope.WebCamBean.imagePhoto}"/>
                        <af:button text="Сделать фото" id="b7">
                            <af:showPopupBehavior popupId="pWebCam"/>
                        </af:button>
                    </af:panelGroupLayout>
                </f:facet>
            </af:panelStretchLayout>
        </af:form>
    </af:document>
</f:view>
```

Сервлет **getphotoservlet** вызывается с фиктивным параметром содержащим время в миллисекундах, для того чтобы обойти проблему с кешированием изображения.